Ingeniería Técnica de Telecomunicaciones:
Especialidad Sonido e Imagen

PROYECTO FIN DE CARRERA

Desarrollo de una aplicación práctica para dispositivos móviles combinando bases de datos y programación en Java

Autor: Efrén Zurita Alonso

Tutor/Director: Dr. David Griol Barres

Leganés, junio de 2015

Título: Desarrollo de una aplicación práctica para dispositivos móviles combinando bases de datos y programación en Java
Autor: Efrén Zurita Alonso
Director: Dr. David Griol Barres

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Un profundo agradecimiento a mi tutor David Griol, por su dedicación y su apoyo a la hora de realizar este proyecto, le agradezco lo que he podido aprender.

Gracias a mis compañeros de Universidad por todos estos años en los cuales hemos realizado juntos la carrera.

Y por último gracias a mis padres, hermanas y el resto de mi familia, los cuales siempre están ahí para apoyarme cuando lo necesito, dándome fuerza y apoyo y animándome cuando tengo peores momentos.

Resumen

El objetivo principal de este Proyecto Final de Carrera es el desarrollo de una aplicación mediante el lenguaje de programación Java que ofrezca un servicio de videoclub virtual en dispositivos móviles. La aplicación desarrollada ofrece las funcionalidades requeridas para agregar y borrar clientes o películas del videoclub, actualizar las bases de datos de clientes y películas, realizar un alquiler por parte de un cliente ya existente, recomendar películas en base a los gustos del cliente, mostrar el día de cierre de los alquileres mensuales, obtener unas listas de todos los clientes, todas las películas, y los alquileres existentes. Todo ello mediante el manejo de ficheros, bases de datos, imágenes, métodos y clases Java. El objetivo de esta aplicación es dar comodidad al usuario/cliente para que pueda acceder fácilmente a este servicio de forma on-line desde el lugar en el que se encuentre, ya que hoy en día, el usuario cada vez mas, busca servicios accesibles desde cualquier punto sin tener que moverse ni trasladarse a otro lugar.

El proyecto fin de carrera se complementa con un análisis detallado de las tecnologías que usamos para la aplicación(J2EE, bases de datos, ficheros, XML)

Palabras clave: Desarrollo software, Java, Bases de datos, Dispositivos móviles, Android.

Abstract

The main objective of this Final Project is to develop an application using the Java programming language that offers a virtual video store service for mobile devices.

The application provides the functionalities required to add and delete clients or movies from the video store, update databases of customers and movies, make a holiday by an existing customer, recommend movies based on customer tastes, show closing day of the monthly rent, get lists of all customers, all movies, and existing properties. These functions are achieved by managing files, databases, images, methods and Java classes. The purpose of this application is to ease the provision of this service from almost everyplace, as today, the user increasingly seeks access all services any point without having to move or move elsewhere.

The project is complemented by a detailed analysis of the technologies used to develop the application (J2EE, databases, files, XML).

Keywords: Software development, Java, Databases, Mobile devices, Android.

Índice general

Índice de Figuras

Índice de Tablas

CAPÍTULO 1. Introducción

El auge de las tecnologías de la información en el mundo actual genera la necesidad cada vez mayor de poder acceder a los datos desde cualquier sitio y a la máxima velocidad posible, hoy en día realizamos compras desde casa, consultamos un sistema GPS para poder ir de un sitio a otro con una ruta específica y porque no vamos a poder comprar una película desde casa para poder verla sin necesidad de acercarse al videoclub a comprarla.

Los ordenadores y los dispositivos móviles se han convertido en una parte esencial de nuestras vidas, accedemos a ciertas aplicaciones y funcionalidades que unos años atrás eran impensables, hoy en día se quiere acceder a estos servicios en cualquier momento y de una forma fácil y eficiente, porque ante todo lo que se busca es comodidad. Los usuarios desean acceder a estas funcionalidades de forma fácil y eficiente desde cualquier lugar y en cualquier momento, por lo que se busca interfaces que establezcan formas más intuitivas de comunicarse entre dichos usuarios y las máquinas.

El uso de las aplicaciones en los dispositivos móviles es una práctica cada vez más extendida, si tenemos en cuenta que hoy en día la gran mayoria de la población cuenta con una Tablet o un SmartPhone para uso personal, nos da una visión del nicho tan grande que tenemos.

En vista de las necesidades del usuario, cada vez mayores en cuanto a flexibilidad y comodidad, mediante el desarrollo de esta pequeña aplicación, se ofrece al usuario la posibilidad de poder acceder de una manera fácil y sencilla a un videoclub.

1.1 Objetivos

Como hemos explicado anteriormente, el objetivo principal de este Proyecto Fin de Carrera es realizar una aplicación en Java[1.1.1] con la que los usuarios finales puedan interactuar de forma

sencilla utilizando sus móviles. La aplicación basada ofrece un servicio de videoclub virtual, en el cual el usuario no tendrá más que ingresar sus datos y realizar el alquiler de una película a través de un dispositivo tecnológico(PC, móvil..) y podrá indicar también la fecha de entrega, el día con el alquiler más alto y más bajo, podrá actualizar los datos y darse de baja de la aplicación, e incluso podrá tener la posibilidad de que la aplicación te recomiende una película conforme a tus gustos. Está aplicación estará desarrollada en Java y posteriormente adaptada para que pueda ser usada desde un dispositivo móvil.

1.2 Antecedentes

En la actualidad, los dispositivos móviles se han convertido en una parte imprescindible en nuestras vidas, trabajamos con ellos, nos comunicamos con ellos y nos hacen acceder a información y funcionalidades las cuales eran impensables hace unos años.

Según los resultados de un estudio publicado por E-innova sobre "El Auge en el uso de aplicaciones móviles de ecommerce t compras online" [1.2.1], en el 2014 el uso de aplicaciones móviles aumento un 76% y las aplicaciones de compras un 174%, por lo que podemos apreciar el uso creciente de estas aplicaciones. Además, se prevé que para el 2015 los dispositivos móviles puedan alcanzar un tercio de todas las transacciones que sean realizadas a través de Internet. Para apreciar la importancia de esto, es conveniente saber que en nuestro país, más del 90% de la población tiene actualmente una Tablet o un Smartphone con el que acceder a todo tipo de aplicaciones (organizar viajes, hacer la lista de la compra).

Según el estudio citado anteriormente, la venta de Tablets superará a la de los PC en 2015, calculando esta cifra en un total de 321 millones de unidades de Tablets, mientras que la venta de PC decrecería de 276 millones de unidades en 2014 a los 262 millones de unidades en 2015. Otro dato a destacar es el porcentaje de ventas de Smartphone representara el 88% de las ventas totales de teléfonos móviles para el año 2018.

El estudio de The App Date [1.2.2] revela que en España el uso de aplicaciones móviles prácticamente es proporcional entre hombres y mujeres, mientras que el 52% de los usuarios de

estas aplicaciones son hombres, el 48% restante son mujeres. Como es de esperar, la generación joven la más activa en cuanto al uso de aplicaciones móviles, el 39% de los usuarios de estas aplicaciones se encuentran entre los 25-34 años de edad y el 26% se encuentran entre los 35-44 años de edad. El horario de actividad en el uso de estas aplicaciones móviles coincide con el final de la jornada laboral, por lo que se puede comprobar que una mayoría de usuarios hacen uso de estas aplicaciones en su tiempo libre, siendo las ciudades de Madrid, Barcelona y Sevilla donde reside el usuario español medio de aplicaciones móviles.

1.3 Material

1.3.1 Material Hardware

El material hardware empleado para la realización del Proyecto Fin de Carrera ha sido el siguiente:

- Ordenador portátil.
- Manual de programación en Java

1.3.2 Material Software

El material software empleado incluye:

- Paquete Office 2007
- Entorno de desarrollo Netbeans
- Editor de texto notepad ++
- Sistema operativos Windows 7

En el apartado de presupuesto especificamos la información sobre el coste de estos materiales que hemos necesitado, en cuanto a la documentación examinada, toda esa documentación se encuentra detallada en el apartado de bibliografía.

1.4 Planificación

La realización de este Proyecto Fin de Carrera se ha estructurado en tres fases diferenciadas que se explican a continuación:

Fase 1: Planificación

- Estudio de la tecnología Java: Aproximación a la tecnología Java.
- Decidir la aplicación: Determinación de la aplicación que se va a implementar y los requisitos que cumplirá la aplicación desarrollada.

Fase 2: Desarrollo

- Análisis y Diseño inicial: División de las distintas funcionalidades de la aplicación en los diferentes módulos y submódulos.
- Implementación de la Aplicación: Desarrollo de todos los módulos y submódulos de la aplicación.
- Añadir Prestaciones: Mejorar la aplicación con nuevas prestaciones que den al usuario una mayor posibilidad de interactuar con la aplicación.
- Reparación de Errores: Proceso de pruebas de la aplicación completa y reparación de los errores que surjan hasta alcanzar una versión estable del proyecto.
- Evaluación de la Aplicación: Estudio de los sistemas de evaluación para aplicaciones Java, realización de las preguntas de la encuesta de evaluación, recogida y análisis de datos.

Fase 3: Documentación

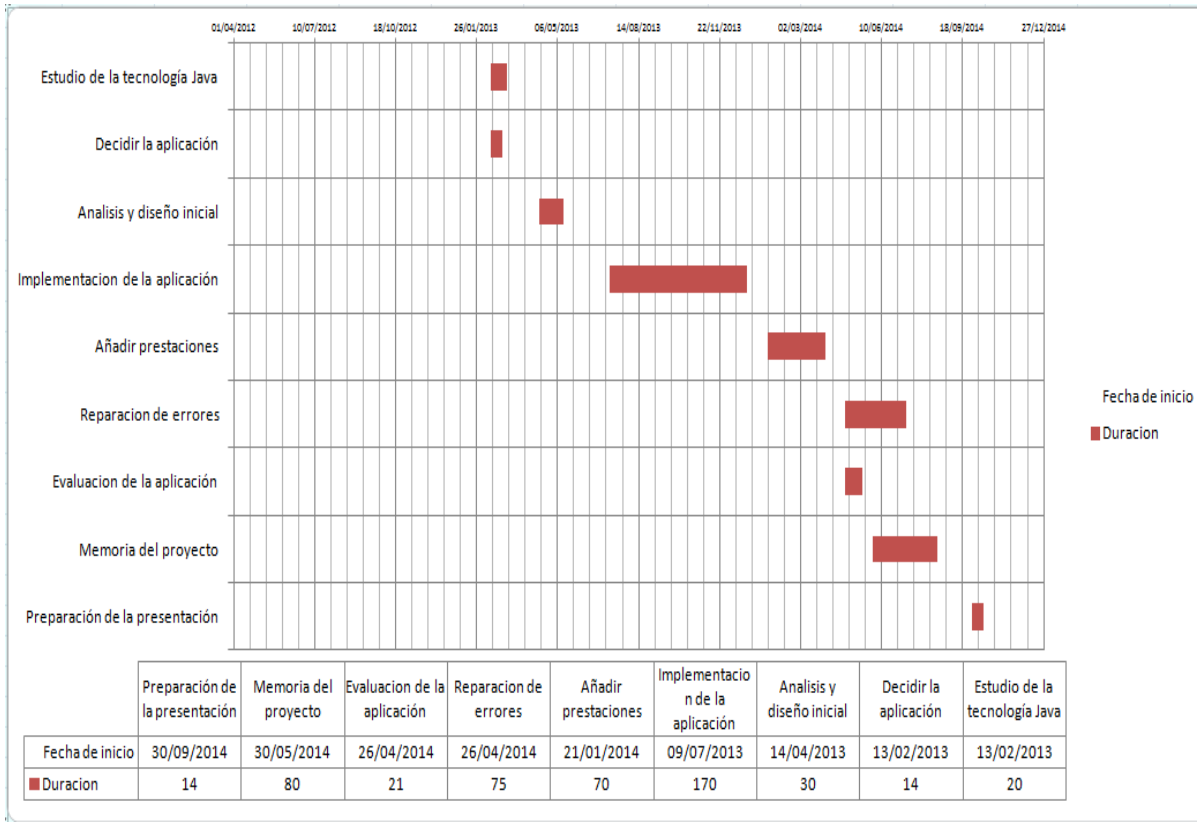
- Memoria del Proyecto: Redacción del presente documento de memoria de Fin de Proyecto.
- Preparación de la Presentación

Una vez establecidas las fases en las que se ha dividido el Proyecto Fin de carrera, se ha

generado un diagrama que posibilite un seguimiento detallado de la evolución de cada tarea.

Para la planificación temporal de las fases del proyecto, se ha usado un diagrama de Gantt, se han incluido las nueve tareas que se citan en el apartado anterior. Se han introducido las duraciones de cada tarea considerando todos los días de la semana laborables y una jornada de 8 horas.

En la Figura 1.1 se puede observar el diagrama de Gantt resultante. Con este diagrama se muestra una visión conjunta de la aplicación del Proyecto Fin de carrera, de sus distintas fases y su duración.



	Preparación de la presentación	Memoria del proyecto	Evaluacion de la aplicación	Reparacion de errores	Añadir prestaciones	Implementacion de la aplicación	Analisis y diseño inicial	Decidir la aplicación	Estudio de la tecnología Java
Fecha de inicio	30/09/2014	30/05/2014	26/04/2014	26/04/2014	21/01/2014	09/07/2013	14/04/2013	13/02/2013	13/02/2013
Duracion	14	80	21	75	70	170	30	14	20

Figura 1.1 Diagrama de Gantt de la planificación temporal del Proyecto Fin de Carrera

1.5 Presupuesto

En esta sección se presenta el presupuesto del proyecto. En dicho presupuesto se va a contemplar la duración de las distintas fases del proyecto y las tareas que hemos realizado e incluimos un desglose de costes de personal y costes de material.

Tareas

Fase 1: Planificación.

- Estudio de la tecnología Java: Lectura de manuales, trabajo con ficheros y bases de datos, interfaces gráficas. Duración: *20 días.*
- Decidir la aplicación a realizar Duración: *14 días.*

Fase 2: Desarrollo.

- Implementación de la aplicación. Duración: 170 días.
- Añadir prestaciones. Duración: 70 días.
- Reparación de errores. Duración: 75 días.
- Evaluación de la aplicación. Duración 21 días.

Fase 3: Documentación.

- Memoria del proyecto Final de Carrera. Duración: 80 días.
- Preparación de la presentación. Duración: 14 días.

Recursos

Para la realización del presente proyecto se han utilizado los siguientes recursos:

• **Recursos software:**

- Skype: **0€**
- Licencia Microsoft Office 2007: **200€**
- Entorno de desarrollo netbeans: **0€**
- Editor de programac Notepad ++: **0€**
- Lenguajes de programacion java: J2EE, J2ME: **0€**

• **Recursos hardware:**

- Dispositivo móvil con plataforma android: 80 €
- Manual de programación en Java: 60€
- Ordenador Portátil: 600€

• **Recursos humanos:**

En la realización de este proyecto han participado dos personas, el director de proyecto y el desarrollador.

Las horas dedicadas al proyecto han sido de una media de 4 horas diarias, contemplando semanas de 7 días.

- Coste de un Ingeniero: 6 €/hora.

Resumen de costes

El coste de recursos humanos de la aplicación se resume en la Tabla 1.1

TAREA	DÍAS	IMPORTE
Fase1	34	816€
Fase2	336	8064€
Fase3	94	2256€
Subtotal	464	11.136€

Tabla 1.1. Detalle de Costes de Recursos Humanos del Proyecto

El coste de recursos humanos de la aplicación se resume en la Tabla 1.2

CONCEPTO	IMPORTE
Recursos Humanos	11.136€
Recursos Software	200€
Recursos Hardware	740€
Subtotal	12076€
(21% IVA)	2536€
TOTAL	14.611 €

Tabla 1.2. Detalle de Coste Total del Proyecto

El presupuesto total de este proyecto asciende a la cantidad de **CATORCE MIL SEISCIENTOS ONCE EUROS**

1.6 Estructura de la memoria

Para facilitar la lectura de esta memoria vamos a continuación a incluir un breve resumen de los capítulos que la integran:

Capítulo 1: Introducción. En este capítulo se establece el propósito de la aplicación que vamos a desarrollar, el objetivo que se quiere conseguir, la planificación del Proyecto Fin de carrera, las fases de desarrollo, los medios utilizados para llevar a cabo la aplicación y la estructura de la memoria

Capítulo 2: Estado del Arte. En este segundo capítulo se lleva a cabo un estudio completo del auge de los dispositivos móviles en la actualidad. Tambien se hace un estudio detallado sobre aplicaciones similares a nuestra aplicación del Proyecto Fin de Carrera que están en el mercado

Capítulo 3: Descripción de la Aplicación. En este capítulo se proporciona una visión global de la aplicación desarrollada. Posteriormente se describe detalladamente cada uno de los módulos de nuestra aplicación.

Capítulo 4: Evaluación de la Aplicación. En este capítulo se lleva a cabo la evaluación de la aplicación mediante una encuesta llevada a cabo a los usuarios de la aplicación. Una vez se recogen los datos, se genera una seria de gráficas que indican la calidad de la aplicación.

Capítulo 5: Conclusiones y trabajo futuro. En este capítulo se exponen las principales cuestiones y conclusiones derivadas de la realización del Proyecto Fin de carrera así como hacia donde se puede encaminar las mejoras a partir de la aplicación resultante.

Glosario : En este apartado se recopilan los principales terminos usados en la memoria para facilitar la lectura del usuario.

Bibliografía : En este apartado se reflejan las citas bibliográficas que se han usado para la realización de la memoria

CAPÍTULO 2. Estado del arte

En este capítulo se analiza el contexto en el que se enmarca el Proyecto Fin de Carrera. En primer lugar, se presenta el entorno de desarrollo utilizado para nuestra aplicación, explicando para qué se utiliza, sus distintas versiones, los lenguajes de programación utilizados con este entorno de desarrollo y como empezar un proyecto nuevo. A continuación, se hace un estudio completo de distintas aplicaciones similares que están en el mercado muy similares a la de nuestro Proyecto Fin de carrera. Por último, se describe de forma detallada como hemos adaptado nuestra aplicación a estas aplicaciones.

2.1 Entorno de desarrollo

Para la realización de este Proyecto Fin de Carrera, se ha usado el entorno de desarrollo **NetBeans**, el cual es un entorno de desarrollo integrado libre hecho para programación java principalmente. NetBeans es libre y gratuito.

El entorno de desarrollo NetBeans soporta el desarrollo de todo tipo de aplicaciones Java:

- Aplicaciones J2SE
- Aplicaciones Web
- Aplicaciones EJB
- Aplicaciones móviles.

NetBeans da soporte a las siguientes tecnologías entre otras: Java, PHP, C/C++, HTML5. NetBeans permite que las aplicaciones sean desarrolladas a traves de una serie de componentes de software llamados módulos. Un módulo es una archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans ademas de un archivo especial, el manifest file, que lo identifica como módulo.

Estas aplicaciones construidas a traves de esos módulos se pueden ir ampliando con otros módulos. Estos módulos se pueden desarrollar de forma independiente por lo que las aplicaciones que se basen en Netbeans pueden ser extendidas por otros desarrolladores de software. Netbeans cuenta con distintas versiones lanzadas desde hace más de una década hasta nuestros días, para esta aplicación se usa la version 6.7.1, en la Tabla 2.1 se aprecian las versiones más recientes.

Versión	Fecha de lanzamiento
NetBeans 8.0.1	5 de octubre de 2014
NetBeans 7.4	15 de octubre de 2013
NetBeans 7.3.1	12 de junio de 2013
NetBeans 7.3	21 de febrero de 2013
NetBeans 7.2	Noviembre de 2012
NetBeans 7.1.2	Mayo de 2012
NetBeans 7.0.1	01 de agosto de 2011
NetBeans 7.0	20 de abril de 2011
NetBeans 6.9.1	4 de agosto de 2010
NetBeans 6.9	15 de junio de 2010
NetBeans 6.8	10 de diciembre de 2009
NetBeans 6.7.1	27 de julio de 2009

Tabla 2.1: Versiones Netbeans

Una vez descargada alguna de las versiones de NetBeans el usuario puede iniciar la aplicación. La Figura 2.1 muestra la ventana que aparece una vez se arranca la aplicación.

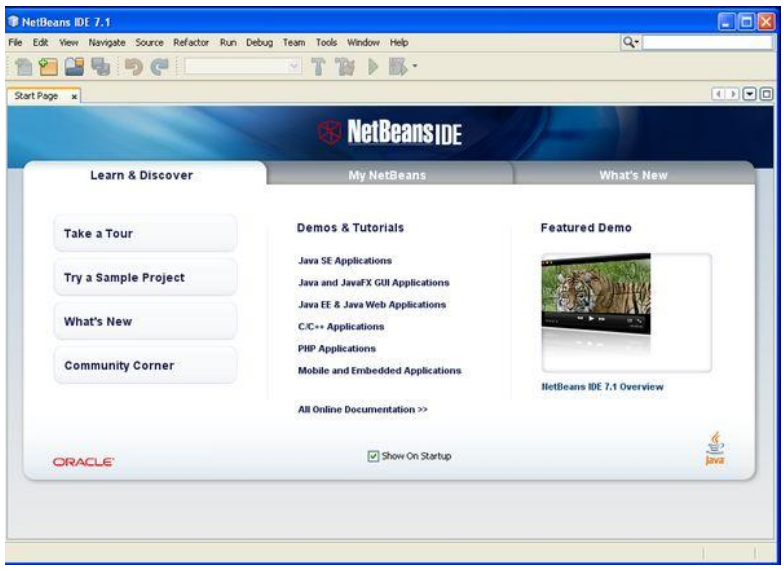


Figura 2.1. Arranque de NetBeans

Como se puede observar en la figura anterior, en la parte superior de la pantalla se aprecian una serie de menús mediante los cuales el usuario podra interactuar con la aplicación para inicializar un proyecto. Esta es la pantalla principal de la aplicación que el usuario visualizará una vez arranque NetBeans.

El menú archivo de la parte de arriba se utiliza para crear un nuevo proyecto tal como se explica a continuación, abrir un proyecto ya existente o importar un nuevo proyecto.

Los menús ejecutar y depurar(hacer un debug) se utilizan para ejecutar la aplicación y depurarla, es decir, poder ir haciendo pruebas para comprobar donde falla la aplicación mediante una serie de trazas o puntos de interrupción que permitan al usuario ir paso a paso hasta encontrar el error. Otra serie de menús que aparecen en la parte superior son : editar, ver, navegar, fuente, restructurar, perfil, equipo, herramientas, ventana y ayuda.

Para iniciar un nuevo proyecto hay que pulsar en el menú archivo situado en la parte superior de la pantalla y posteriormente seleccionar un Proyecto nuevo. Desde este asistente el usuario puede seleccionar la categoria del proyecto a realizar, es decir, el lenguaje de desarrollo que el usuario desee utilizar en el proyecto. Como se explica anteriormente se puede inicializar un proyecto Java, C/C++, Phb, JavaWeb y mas lenguajes. En la Figura 2.2 se muestra el asistente que se utiliza para configurar el proyecto que se vaya a crear. Una vez seleccionado se pulsa en la pestaña siguiente.

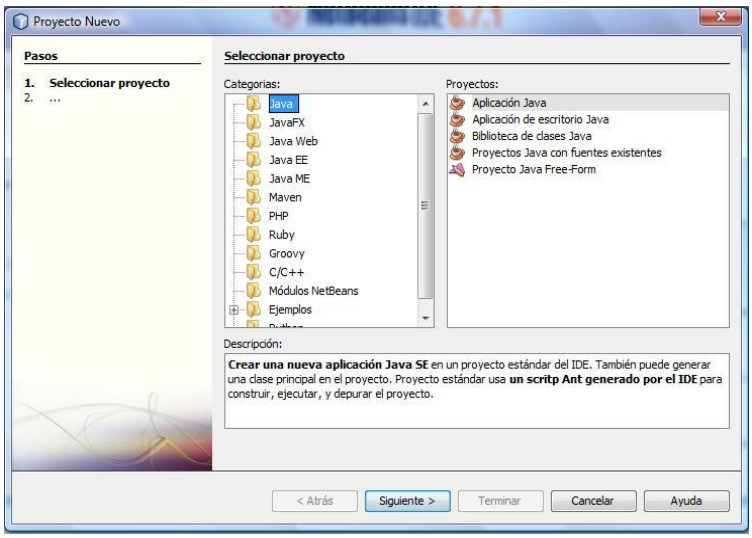


Figura 2.2. Creación del nuevo proyecto

La última parte del asistente de creacion del nuevo proyecto sirve para darle nombre al nuevo proyecto, seleccionar la ubicación del proyecto, crear una carpeta de bibliotecas y elegir la clase principal de este nuevo proyecto. Todo esto queda reflejado tal y como se muestra en la Figura 2.3.

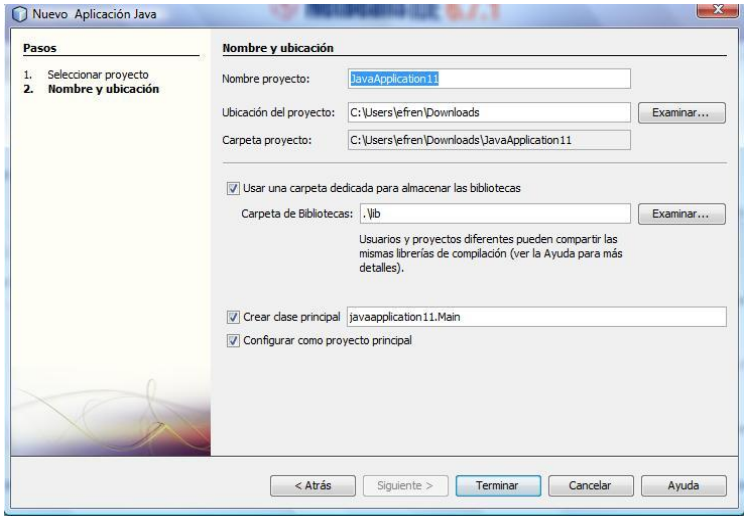


Figura 2.3. Finalizar asistente para la creación del proyecto

Para concluir el usuario pulsara el botón terminar y ya tendra creado su nuevo proyecto. Tal y como se muestra en la Figura 2.4 en la parte central de la pantalla tendremos la clase principal del proyecto. En la parte de arriba se colocan las diferentes pestañas que se podrán seleccionar para elegir una u otra clase en funcion de cómo se vaya a trabajar. En la parte izquierda de la pantalla se aprecia el numero de proyectos que el usuario tiene creados, los paquetes de clases usados en cada uno de ellos, sus librerias y las clases utilizadas.

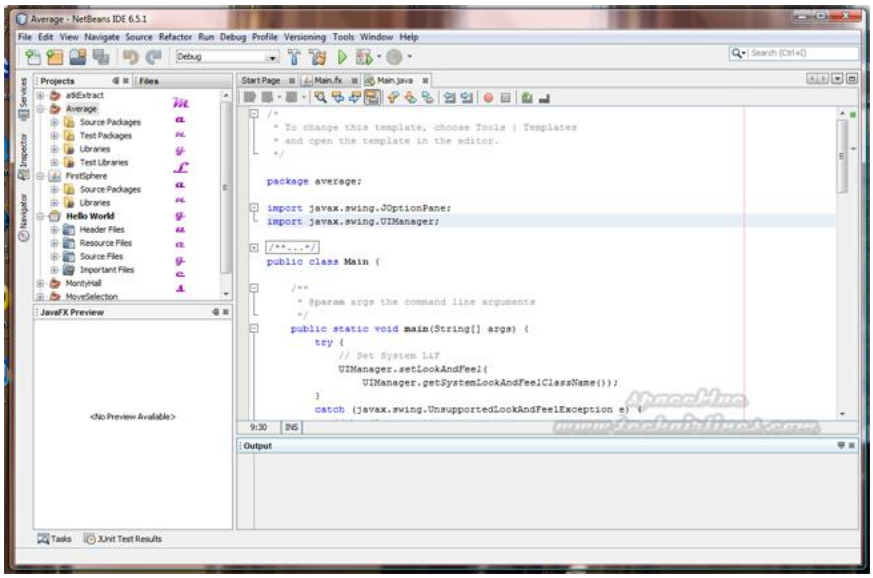


Figura 2.4. Nuevo proyecto creado

2.2 Sistemas operativos móviles

Uno de los ejemplos mas claros de cómo se han acentuado en los últimos años los avances tecnológicos es la telefonía móvil. Los teléfonos móviles se han ido convirtiendo con el tiempo en parte indispensable para la vida diaria de las personas y, gracias a la innovación han pasado a ser pequeños ordenadores de bolsillo para el usuario capaces de hacer todo tipo de tareas sin necesidad de otro tipo de dispositivos.

Hoy en dia estos dispositivos móviles funcionan como localizadores de satélites, centros multimedia, ademas de cómo teléfonos. Gracias a un sistema operativo (SO) que llevan incorporado, pueden realizar estas tareas tan distintas y complejas.

Un sistema operativo es un conjunto de programas que permiten comunicarse al usuario y al hardware del dispositivo gestionando sus recursos de una forma eficiente y comoda. El SO se compone de una serie de capas:

- **Kernel:** Tambien llamado núcleo, proporciona el acceso a los elementos del hardware del dispositivo. Ofrece servicios a las capas superiores tales como los drivers para el hardware, la gestion de procesos, el sistema de archivos, el acceso y la gestión de la memoria.
- **Middleware:** Es el conjunto de módulos que posibilitan la existencia de aplicaciones para móviles. Los servicios que ofrece son el motor de mensajeria y comunicaciones, codecs multimedia, interpretes de páginas webs, gestión del dispositivo y seguridad.
- **Entorno de ejecucion de aplicaciones:** Es un gestor de aplicaciones y un conjunto de interfaces abiertas y que se pueden programar para la creación de software.
- **Interfaz de usuario:** Facilitan la interación con el usuario y el diseño gráfico de la aplicación. Incluye botones, pantallas, listas etc..

2.2.1 Sistemas operativos móviles mas importantes

Hoy en día, el sistema operativo sobre el que trabaja el terminal es una elección muy importante para el usuario, a la hora de poder escoger un smartphone el usuario decide qué sistema operativo le va a ser mas útil en funcion de lo que ofrece cada uno. A continuación se enumeran los principales sistemas operativos para dispositivos móviles con una breve descripción de cada uno de ellos.

iOs

Desarrollado por la empresa Apple Inc. cuyo lanzamiento data de junio de 2007. Fue pensado exclusivamente para el iPhone, siendo despues usado en el iPod Touch e iPad. Su simplicidad y optimización son los pilares que hacen que millones de usuarios se decanten por este sistema operativo en lugar de otras plataformas que requieras mas software.

Este sistema operativo tiene cuatro capas de abstraccion: la capa del núcleo del sistema operativo, la capa de servicios principales, la capa de medios de comunicación y la capa de

Cocoa Touch.

El sistema operativo ocupa menos de medio gigabyte del total del dispositivo, que puede ser de 8 o 16 Gb. Gracias a esto soporta aplicaciones Apple y otras publicadas en la App Store [2.2.1.1].

El lugar en el que se descargan las aplicaciones iOs es la AppStore que es una de las tiendas de aplicaciones más conocidas y de mayor calidad que hay en el mercado.

Una desventaja de este sistema operativo es las pocas posibilidades que hay de cambiar la forma de funcionar un teléfono, existe un control muy rígido de las aplicaciones publicadas debido a que el sistema de Apple es cerrado. Sólo se puede hacer uso de este sistema operativo con los dispositivos citados con anterioridad iPod Touch e iPad por lo que al haber sólo un fabricante y modelo el precio de estos dispositivos es bastante elevado.

Android [2.2.1.2]

Es sin duda el sistema operativo líder en el mercado de sistemas operativos. Esta basado en Linux y diseñado para dispositivos móviles con pantalla táctil. Desde el año 2005 pertenece a Google debido a que fue modificado para ser utilizado en dispositivos moviles como los teléfonos inteligentes y luego en tablets como el caso de Galaxy Tab. De Samsung.

En el año 2007 tuvo lugar la presentación oficial de Android. Desde el primer móvil que se vendió hacia el año 2008 hasta nuestros días su sistema operativo ha sufrido diversas actualizaciones. Cada nueva plataforma que se lanza es compatible con sus versiones anteriores, por lo que solo añaden funcionalidades

Como se ha citado anteriormente, esta basado en Linux, disponiendo de un kernel en este sistema y utilizando una maquina virtual sobre este kernel que es la responsable de convertir el código escrito en java de las aplicaciones a código capaz de comprender el Kernel.

Las aplicaciones para android son desarrolladas en Java con unas APIS propias por lo que las aplicaciones en Java para PC no son compatibles con este sistema.

Debido a la gran cantidad de fabricantes que lo utilizan es mas difícil actualizar los dispositivos Android que los que funcionan con el sistema operativo Apple.

Windows Phone

Anteriormente llamado Windows Mobile se trata de un sistema operativo desarrollado por Microsoft cuyo lanzamiento tiene lugar en octubre de 2010. Su última versión es la versión Windows Phone 10.

Se basa en el nucleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas. Diseñado para ser similar a las versiones de windows en su estética ofrece una gran oferta de software a terceros disponible para Windows Mobile.

Al igual que iOS soportaba aplicaciones de terceros publicadas en la AppStore, existe una demanda de software de terceros disponible para Windows Phone, que se puede adquirir a

través de Windows Marketplace [2.2.1.3] gestionado por microsoft y que incluye un proceso de aprobación por cada aplicación de terceros, en las que se evaluan diferentes características.

WindowsMarket esta disponible en mas de 50 países y el número de apps se ve incrementada diariamente. El proceso de suscripción a esta plataforma conlleva registrarse en el App Hub y cuesta unos 99 dolares. El desarrollador podra publicar hasta un número ilimitado de aplicaciones de pago pero no un número superior a cinco aplicaciones gratuitas.

La variedad de móviles de Windows Phone no es tan amplia como la de otros sistemas operativos como Android, aun así marcas como LG Samsung o HTC incorporan algunos de sus dispositivos móviles.

BlackBerry OS

Es un sistema operativo desarrollado por la empresa canadiense RIM(Research In Motion) para sus dispositivos. Su desarrollo se remonta a la aparición de las primeras PDAs en 1999. No fue hasta 2008 con el lanzamiento de BlackBerry OS 5 cuando se afianzó en los principales sistemas operativos móviles.

La última versión lanzada por este sistema operativo BlackBerry 10 cuenta como principal novedad con la incorporación de un teclado virtual. La tienda BlackBerry World incluye aplicaciones compatibles con BlackBerry 10.

El SDK y el simulador tal y como ocurre en Android son gratuitos por lo que los desarrolladores pueden crear aplicaciones para BlackBerry sin tener que pagar por ello. Para que dichas aplicaciones sean publicadas en la tienda oficial necesitan firmar digitalmente sus aplicaciones para asociarlas a una cuenta de desarrollador de RIM.

Muy pocos dispositivos incorporan este sistema operativo. Algunas marcas como Siemens, HTC y Sony Ericsson utilizan para algunos modelos el cliente de correo electrónico de BlackBerry.

Symbian

Es un sistema operativo producto de la alianza de varias empresas de telefonía móvil en 1997. Estas empresas son Nokia como la mas importante, Sony Ericsson, Samsung, Siemens, BenQ, Fujitsu, Lenovo, Motorola, LG, Phanasonic, Sharp. Esta alianza le permitio por un momento ser uno de los pioneros y mas usados. El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el SmartPhone de Microsoft.

El sistema operativo Symbian es una colección compacta de codigo ejecutable y varios archivos la mayoria de ellos bibliotecas DLL y otros datos requeridos (archivos de configuracion, de imágenes, etc.).

Las aplicaciones se distribuyen a través de la Ovi Store [2.2.1.4], la tienda oficial de descarga de aplicaciones y a diferencia de lo que ocurre con los desarrolladores de aplicaciones para iOs y WindowsPhone, el pago se realiza de una vez y no anualmente. Esta tienda oficial cuenta con mas de 50000 aplicaciones para descargar y mas de 6,5 millones de descargas diarias.

Este sistema operativo esta disponible en una gran cantidad de dispositivos móviles. Destacan Siemens, Panasonic, Lenovo, LG, Samsung, SonyMobile, Communications aunque la mayoria de los dispositivos que soportan el sistema operativo son de la compañía Nokia.

2.2.2 Cuota de mercado Sistemas operativos móviles

Otro aspecto fundamental a la hora de comparar plataformas móviles es su cuota de mercado. En la Figura 2.5 se muestra el mercado de Smartphone por sistemas operativos durante los últimos cinco años [2.2.2.1].

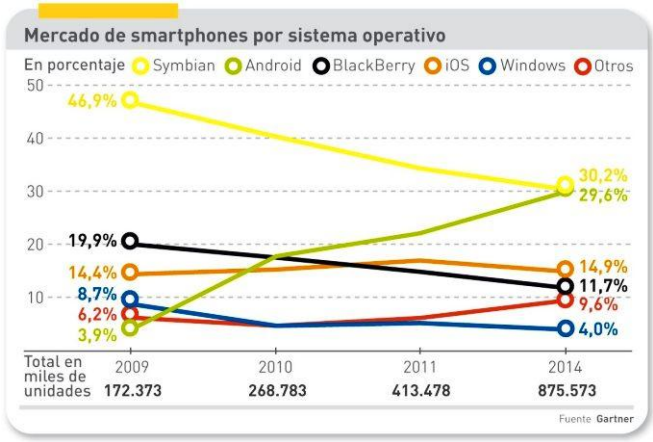


Figura 2.5 Mercado de Smartphone por SO

Como se muestra en la Figura anterior, el mercado mundial de sistemas operativos para dispositivos móviles será dominado en 2014 por Symbian y Android, sumando juntos casi el 60% de las ventas según la firma de investigación de mercado Gartner.

Otra manera de analizar los datos seria ver la evolución de Symbian, pasando en cinco años de consolidar un 47% de cuota de mercado a perder un 17% hasta el 30% llegando a un empate técnico con Android.

Otra lectura de los datos seria constatar la casi desaparición de WindowsPhone del mercado respecto a otras grandes marcas, debido a que en cinco años, en plena reinvención, pasaría del 8,7% al 4% actual. De todos modos, Symbian tiene garantizado el liderazgo, lo que demuestra la fuerza de Nokia todavía.

2.2.3 Aplicaciones móviles

El mercado de las aplicaciones móviles no para de crecer, la oportunidad de negocio es muy grande y las empresas no quieren dejar pasar esa oportunidad. El usuario busca sencillez y comodidad, poder hacer la lista de la compra, alquilar una película del videoclub o disponer de un localizador GPS son algunos de los ejemplos que hacen la vida más fácil al usuario.

Las aplicaciones móviles pueden ser de tres tipos:

- **App Nativas**
- **Web App**
- **Web App Nativa**

App Nativas

Una aplicación nativa es la que se desarrolla de forma específica para un determinado sistema operativo, llamado SDK o Software Development Kit. Para que la app esté disponible en varias plataformas se deben crear varias apps con el lenguaje del sistema operativo seleccionado debido a que Android, iOS y WindowsPhone tienen un sistema diferente. Las aplicaciones para iOS se desarrollan con el lenguaje Objetive-C, las aplicaciones para Android en Java y las de Windows Phone en .net.

Cuando hablamos de desarrollo móvil, la mayoría de las veces nos referimos a aplicaciones nativas. Las aplicaciones nativas pueden funcionar sin conexión a internet. La descarga e instalación de estas aplicaciones siempre se realiza a través de los app store de los fabricantes.

En función del coste o rentabilidad la mejor opción será el desarrollo de una aplicación nativa para cada plataforma a no ser que tu presupuesto sea más limitado, entonces las aplicaciones web serian una buena opción también. La Tabla 2.2 muestra las ventajas e inconvenientes de las App Nativas.

Ventajas	Inconvenientes
✓ Mejor experiencia del usuario ✓ Visibilidad en App Store ✓ Envío de notificaciones a los usuarios ✓ Acceso completo al dispositivo ✓ La actualización de la app es constante	❖ Diferentes idiomas/ herramientas para cada plataforma de destino ❖ Más caras a desarrollar ❖ Código no reutilizable entre plataformas.

Tabla 2.2: Ventajas e Inconvenientes de las App Nativas

Web App

Una aplicación web es desarrollada por lenguajes como HTML, CSS y Javascript. Estas aplicaciones se ejecutan dentro del propio navegador web del dispositivo a través de una URL. El contenido se adapta a la pantalla.

La principal ventaja respecto a la app nativa es la posibilidad de programar independientemente del sistema operativo en el que se use la aplicación, se pueden ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones.

A diferencia de una app nativa no necesita instalación por lo que no pueden estar visibles en la app store y la promoción se debe realizar de forma independiente.
Cuando nuestro objetivo sea adaptar la web al formato móvil las web app móviles serán muy buena opción. La Tabla 2.3 muestra las ventajas e inconvenientes de las Web App.

Ventajas	Inconvenientes
✓ Proceso de desarrollo más sencillo y económico ✓ El mismo código base reutilizable en varias plataformas ✓ No necesitan aprobación externa para publicarse ✓ El usuario siempre dispone de la última versión ✓ Pueden reutilizarse sitios ya diseñados	❖ Acceso muy limitado a los elementos y características del hardware del dispositivo ❖ El tiempo de respuesta y la experiencia del usuario es menor que en una app nativa ❖ Requiere mayor esfuerzo en promoción y visibilidad. ❖ Requiere conexión a internet

Tabla 2.3: Ventajas e Inconvenientes de las Web App

Web App nativa

Una aplicación web app nativa es una combinación de las dos anteriores, es decir, una aplicación hibrida que recoge lo mejor de cada una de ellas. Por una parte se desarrollan con lenguajes propios de las web app como son HTML, CSS, Javascript por lo que permite su uso en diferentes plataformas, pero también dan la posibilidad de acceder a gran parte de las características hardware del dispositivo. La gran ventaja de estas aplicaciones es que a pesar de estar desarrollada con estos lenguajes, es posible agrupar los códigos y distribuirla en la app store.

Varios de los framework más utilizados por los programadores para el desarrollo multiplataforma de aplicaciones hibridas es PhoneGap o Cordova. La Tabla 2.4 muestra las ventajas e inconvenientes de las Web App nativa.

Ventajas	Inconvenientes
✓ Es posible distribuirlas en las tiendas de iOS y Android. ✓ Instalación nativa pero construida con Javascript, HTML y CSS ✓ El mismo código base para múltiples plataformas. ✓ Acceso a parte del hardware del dispositivo	❖ Experiencia del usuario más propia de la aplicación web que de la aplicación nativa. ❖ Su diseño no siempre está relacionado con el sistema operativo en el que se muestre.

Tabla 2.4: Ventajas e Inconvenientes de las Web App nativa

Una vez analizadas el tipo de aplicaciones móviles que hay habrá que seleccionar una u otra en función de las ventajas e inconvenientes de cada una siempre teniendo en cuenta el coste a asumir, las funcionalidades del dispositivo, hacia qué público nos dirigimos, el tipo de diseño, si se desean incluir notificaciones en la app y la previsión a modificar la aplicación o modificarla en un futuro.

Como se ha citado anteriormente, el uso de aplicaciones móviles está creciendo con el tiempo y el usuario cada vez más lo que busca es la comodidad y flexibilidad, poder acceder a determinados servicios de una forma fácil y sencilla. Sentarte en el sofá y escoger que ver y a la hora que tú digas no es cosa del futuro sino del presente. Existen varias páginas web que ofrecen video bajo demanda, con lo que el espectador no tiene que seguir la parrilla de un determinado canal sino que puede ver una película o una serie seleccionándola desde su SmartTV, ordenador o incluso teléfono móvil.

El usuario de estas tecnologías lo que busca es poder elegir que ver en ese momento en función de sus gustos y particularidades. Hoy en día, la tecnología actual tiende hacia un mercado más selectivo, el usuario encuentra la posibilidad de escoger lo que quiere ver y no solo lo que le ofrece en ese momento la parrilla de televisión.

En España, la televisión a la carta gana terreno a la televisión tradicional, tal como podemos encontrar en estos artículos [2.2.3.1] y la tendencia es que se amplié la diferencia.

En la actualidad existen una gran variedad de servicios de video bajo demanda orientados a todo tipo de público, en definitiva, tener el videoclub en casa, lo que ofrece al usuario comodidad y facilidades para poder alquilar esa película que por unas causas u otras no pudo ir a ver al cine. Si bien en España tardaron un poco más en llegar, hoy en día podemos echar un vistazo a las principales alternativas y analizar lo que nos ofrece cada una de ellas.

A continuación, se enumeran una serie de aplicaciones ya existentes que ofrecen un servicio similar al de la aplicación del Proyecto Fin de Carrera.

Youzee [2.2.3.2]

Es un servicio de video bajo demanda, uno de los más comunes actualmente, digamos que Youzee es lo más parecido a tener el videoclub en casa. El registro al entrar a la aplicación es gratuito y una vez dentro del sistema tenemos dos opciones, la primera es pagar por aquellos contenidos que queramos ver, en cuyo caso tendremos un mes para verlos o 48 horas desde el primer play, la segunda es optar por pagar una suscripción mensual de 6,99 euros que nos da acceso a todos los contenidos.

Cabe destacar el **sistema de recomendaciones basado en nuestros gustos** y la posibilidad de ver la valoración que han hecho otros usuarios de esos contenidos. Información muy útil cuando no estamos seguros de qué vamos a ver exactamente.

Wuaki.tv [2.2.3.3]

Es una plataforma española, ofrece un servicio muy similar a Youzee, pero con una serie de diferencias: no ofrece un modelo de suscripción y ofrece la posibilidad de no solo alquilar películas sino de comprarlas.

Los precios de las películas varían de unas a otras y en función de la calidad de video (HD o SD), una vez realizado el pago y a diferencia de Youzee tendremos 48 horas para verlas todas las veces que queramos.

Al igual que ocurre con Youzee el catálogo no es excesivamente amplio. Wuaki cuenta con una serie de contenidos totalmente gratuitos, gracias a los cuales permite ver el rendimiento del servicio sin pasar por caja.

Cineclick[2.2.3.4]

Cineclick es uno de los servicios que más tiempo lleva operando en nuestro país. Fue creada en 2007 y está disponible en versión web, para móviles y tablets y también para la SmartTV de tu televisor.

Tiene un catálogo de más de 1000 títulos a los que se puede acceder a través de su tarifa plana de 9,95 euros al mes o por alquiler individual.

Filmin[2.2.3.5]:

Es uno de los proyectos de video bajo demanda que mas lleva en España aunque esta más orientado a cine independiente. Su estructura es similar a Wuaki, tiene una tarifa plana de 8 euros al mes para ver cierta parte del catálogo y estrenos que se pueden alquilar a partir de 2 y 3 euros, aunque todavía no está disponible en la SmartTV de Philips.

Filmin ofrece un catálogo con 4493 películas y 136 series, recomendado especialmente para personas que les guste el género independiente y los documentales.

CAPÍTULO 3. Descripción de la aplicación desarrollada

En este capítulo se describen las características generales del conjunto de módulos que conforman nuestra aplicación: funcionalidad y arquitectura.

3.1Presentación de la aplicación

La aplicación desarrollada para este Proyecto Fin de Carrera con la tecnología Java J2EE ofrece al usuario un conjunto de funcionalidades que se enumeran a continuación. Dichas funcionalidades harán posible el acceso del usuario al videoclub de una forma fácil y cómoda.

Figura 3.1. Presentación de la Aplicación

Como se puede observar en la Figura 3.1, al abrir la aplicación aparece esa pantalla de presentación con una serie de menús en la parte de arriba mediante los cuales el usuario interactuará con la aplicación para realizar las operaciones que desee en cada momento.

Como se puede comprobar en la barra de menús situada arriba, se cuenta con los menús Sistema, Actualiza, Movimientos y Listados, el detalle de los cuales se profundiza más adelante.

El menú Sistema tiene los submenús:

- **Acerca de**, indica la versión de la aplicación, el nombre y apellidos del autor del Proyecto Final de Carrera, el título de la carrera realizada por el autor y unos datos más abajo referidos a el PC con el que se abre la aplicación, como el usuario, el equipo, el sistema operativo, el procesador y la versión JDK utilizada en ese procesador.
- **Salir**, sirve para salir de la aplicación.

El menú Actualiza tiene los submenús:

- **Ingresar,** se utiliza para añadir clientes o películas a la base de datos de nuestro videoclub.
- **Borrar**, se utiliza para borrar clientes o películas de la base de datos del videoclub.
- **Actualizar**, se utiliza para actualizar los datos de un cliente o una película, ya sea añadir algún dato nuevo o modificar un dato erróneo.

El menú Movimientos consta de estos submenús:

- **Alquiler de una película,** se selecciona la película que se quiera alquilar.
- **Cierre mensual de alquileres**, se informa de todos los movimientos acontecidos durante ese mes, los ingresos obtenidos en ese mes, toda la información referida a los clientes y las películas que se han alquilado, las fechas en las que ese cliente a alquilado la película, el coste de cada alquiler. Todo ello vendrá incluido en un informe
- **Recomendación de Películas**, se tendrá de un formulario a rellenar y en función de los gustos personales de cada persona, edad y otros aspectos se recomendará una película.
- **Cartelera,** se accederá a la cartelera**.**
-

El menú Listados consta de estos submenús:

- **Listado de todas las películas**
- **Listado de todos los clientes**

- **Listado de las películas por cliente**, un listado en el que aparece el número de alquiler y las películas alquiladas por cada cliente.
- **Día del mes con el alquiler más alto y el más bajo**, para cada mes en una tabla aparece el día que se ha realizado el alquiler más alto y el más bajo.

3.2 Diseño de la aplicación

La Figura 3.2 muestra el diseño de los menús que tendrá la aplicación para interactuar con el usuario.

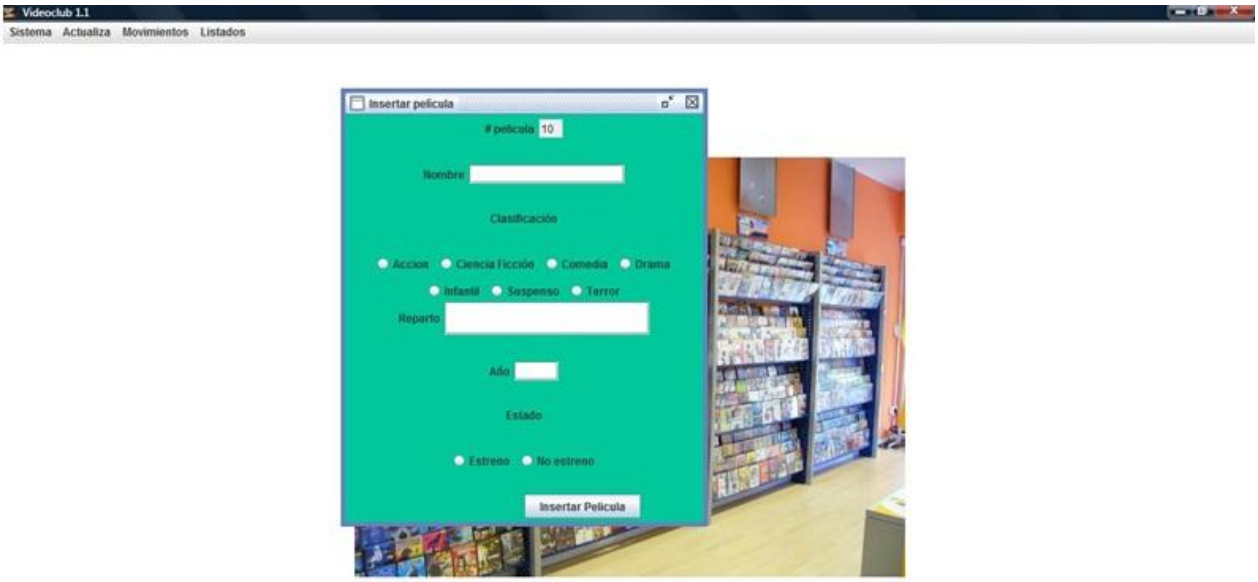


Figura 3.2. Diseño de la Aplicación

En esta aplicación de nuestro Proyecto fin de carrera lo que se busca es mostrarle al usuario una "*interfaz gráfica*" lo más amena posible, es decir construir unos mecanismos para construir ventanas, botones, menús, etc. que nos permitan crear una interfaz amigable para nuestra aplicación. Los mecanismos para crear "*interfaces de usuario*" en Java están pensados para favorecer la creación de la lógica de negocio separada de la creación de la interfaz de usuario, sin embargo, no hay garantía de que esto suceda, es nuestra responsabilidad como programadores cristalizar este objetivo.

La aplicación está diseñada de un modo que facilite las cosas al usuario, así como a través de una interfaz vistosa en las que el usuario pueda sentirse cómodo ya que está pensado para que la utilicen usuarios de diferentes condiciones y/o edades.

El diseño consta de una pantalla principal explicada anteriormente con una barra de menús en la parte de arriba, en cada uno de los cuales aparecen varios submenús que realizan diferentes operaciones, cuando pulsamos alguno de esos submenús como el de agregar una película a la base de datos existente en la aplicación, nos aparece una ventana o panel como el que aparece en la figura de arriba donde en la parte superior se tiene el título de la gestión que estemos realizando y se nos presentará una ventana con un formulario para que el usuario vaya rellenando los datos de la manera más cómoda posible todo ello aderezado con un diseño vistoso en cuanto a colores y forma. En el formulario aparecen etiquetas de texto, radio botones, áreas de texto y todo lo necesario para dar al cliente el mayor manejo de la aplicación, por último hay una serie de botones en cada uno de estos paneles que sirven para realizar las funciones definidas por cada submenú de la aplicación.

3.3 Módulos de la aplicación

Nuestra aplicación contiene las siguientes clases dentro de un paquete de fuentes llamado videoclub1:

- ✓ Principal.java
- ✓ pintaImagen.java
- ✓ panelImagen.java
- ✓ AcercaDe.java
- ✓ Seleccionpelicula.java
- ✓ actualizarClientes.java
- ✓ actualizarPelicula.java
- ✓ alquilerDePelicula.java
- ✓ diaAlquilerAltoBajo.java
- ✓ borrarClientes.java
- ✓ borrarPelicula.java
- ✓ cierreMensual.java
- ✓ Recomendación.java
- ✓ insertarClientes.java
- ✓ insertarPelicula.java
- ✓ listadoClientes.java
- ✓ listadoPelicula.java
- ✓ listadoPeliculasCliente.java
- ✓ informeCierreMensual.java
- ✓ VerCartelera.java

En la Figura 3.3 se puede ver la declaración de la clase Principal.java, dicha clase tiene la funcionalidad de dotar la estructura genérica de la pantalla de inicio.

```
public class principal  extends JFrame implements ActionListener{


    AcercaDe AD;
    insertarClientes ingresaClientes;
    insertarPelicula ingresaPelicula;
    alquilerDePelicula alquiler;
    borrarClientes borrarCli;
    borrarPelicula borrarPel;
    actualizarClientes actualizarCli;
    actualizarPelicula actualizarPel;
    cierreSemanal cierreSem;
    cierreMensual cierreMen;
    listadoClientes listadoCli;
    listadoPeliculas listadoPel;
    listadoPeliculasCliente listadoPPC;
    diaAlquilerAltoBajo alquilerDAB;
    pintaImagen pim;
    static Selecionpelicula seleccionpel;
    static informeCierreMensual infoCierreMensual;

    JDesktopPane JDP_escritorio;
    JMenuBar JMB_menu;
     /* Barras de menú principales*/

    JMenu      JM_sistema, JM_actualizar, JM_movimientos, JM_listados,

    /*Barras de menú secundarias*/

    JM_ingresar, JM_borrar, JM_atualizar  ;
```

Figura 3.3. Declaración clase principal

Como se puede ver en la clase principal.java se realizan las instancias de las demás clases.java que se utilizan en la aplicación. Se crea además el JDesktopPane, que contiene el panel principal, y el JMenuBar, que contiene el menú principal y los distintos menús que están dentro del JMenuBar:

- Sistema
- Actualizar
- Movimientos
- Listados

A continuación se crean los menús secundarios y se irán agregando cada submenú en los menús principales y así con todos los objetos de la interfaz gráfica.

En la Figura 3.4 se muestra como desarrollamos el método actionPerformed para la clase principal.java, y de esta forma añadimos la funcionalidad al JMenuItem

```
    JMI_salir.addActionListener(new ActionListener() {  //
        private int YES_OPTION;
        private int YES_NO_CANCEL_OPTION;

    /*Pulsamos para salir de la aplicacion*/

 public void actionPerformed(ActionEvent e) {
if(JOptionPane.showConfirmDialog(null,"¿Desea salir?" , "Salir", YES_NO_CANCEL_OPTION)==YES_OPTION)
 System.exit(0);  }});

  // Se llama a la clase AcercaDe

  JMI_AcercaDe.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent e) {

          AD=new AcercaDe();
          AD.setResizable(false);
          if(siJIF_EstaAbierto("Acerca de")!=true)
          JDP_escritorio.add(AD);

        }
     });

    //se asigna la clase ActionListener al Item ingresar clientes del menú de actualiza. Se llama a la clase ingresarClientes

    JMI_ingresarClientes.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent e) {
      ingresaClientes =new insertarClientes(null, null, null, null, null, null);
      ingresaClientes.setResizable(false);
      if(siJIF_EstaAbierto("Insertar Cliente")!=true)// si el cliente no existe le añadimos
```

Figura 3.4. Clase Principal

Desde el método addActionListener () se da funcionalidad a un botón o a un JMenuItem, este método necesita que le pasemos una clase que implemente la interfaz ActionListener().
La interfaz ActionListener () contiene el método ActionPerformed (). Para evitar problemas al compilar se tiene que definir el método ActionPerformed().

En este método ActionPerformed() básicamente lo que consigue es añadir como escuchador a cada JMenuItem de tal manera que cada vez que se pulse sobre un menú aparezca en la pantalla, para eso se comprueba si el JItemFrame está abierto, si no lo está se añade la instancia de cada clase al JDesktopPane.

En la Figura 3.5 se muestra como está creado el método Main de nuestra aplicación.

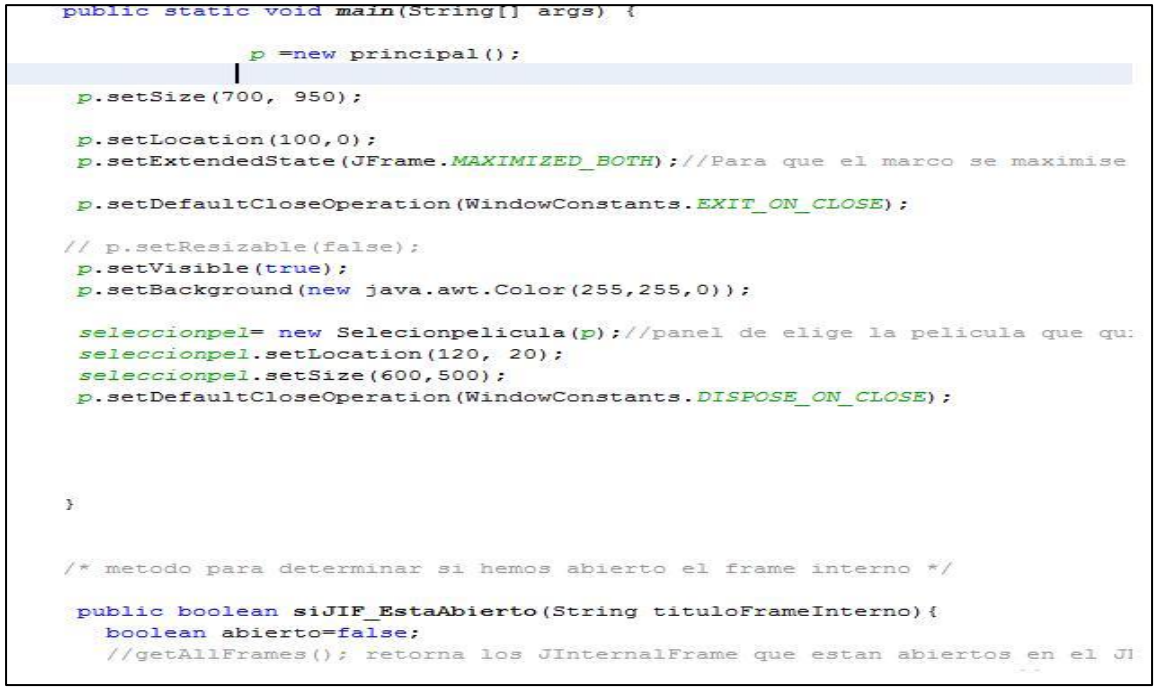


Figura 3.5. Método Main

En Java existe un método especial llamado Main que es el punto de partida de nuestro programa, dentro de este método se pueden hacer las llamadas a todas las rutinas que componen nuestro programa. El método principal Main de una Clase Java es inalterable, es decir:

- Siempre debe incluir los calificadores: public y static
- Nunca puede retornar un valor como resultado, es decir, siempre debe indicar el valor void como retorno.
- Su parámetro de entrada siempre será (String []) el cual es tomado de la línea de comandos o una fuente alterna.

  Aunque no es un requerimiento definir el método Main dentro de toda Clase Java, dicho método representa el único mecanismo automático para realizar tareas al invocarse una Clase, es decir, al momento de ejecutarse determinada Clase siempre será ejecutado todo el contenido dentro de dicho método.

3.3.1 Sistema

El menú Sistema consta de dos submenús, uno para salir de la aplicación y otro meramente informativo que es el menú Acerca de.

3.3.1.1Acerca de

La Figura 3.6 muestra la ventana que aparecerá al pulsar el submenú Acerca de.

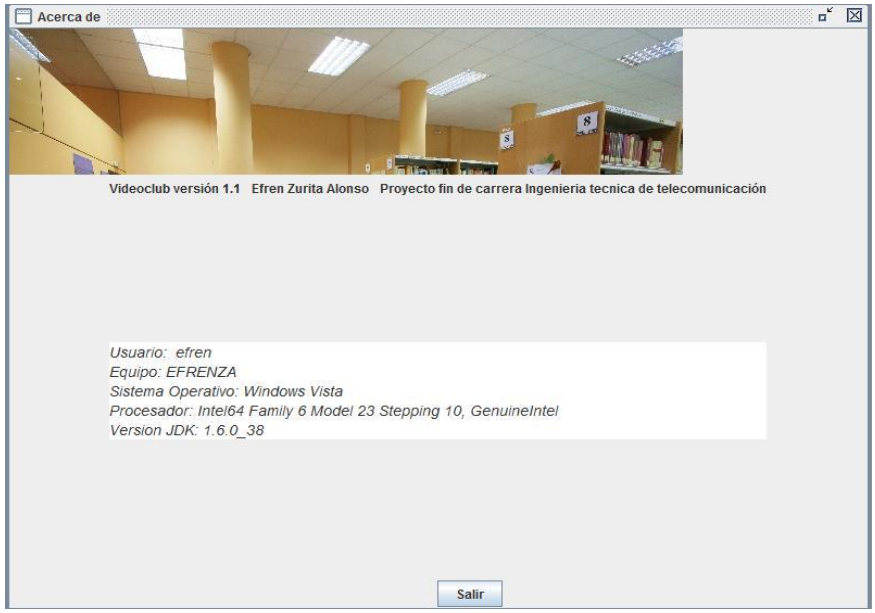


Figura 3.6. Acerca De

Esta sencilla ventana muestra el nombre de la aplicación desarrollada para el Proyecto Fin de carrera, junto con la versión actual, el nombre y apellidos del autor, la carrera que ha cursado el autor y en la parte inferior de la ventana se pueden ver unos datos referentes al sistema en el que se ha creado la aplicación como puede ser el nombre del usuario, el equipo en el que estamos ejecutando la aplicación , el sistema operativo de nuestro equipo, el procesador del equipo en el que estamos desarrollando la aplicación y la versión del JDK que estamos usando en nuestro procesador.

En la Figura 3.7 puede apreciar el código que se usa para la implementación de este submenú Acerca de.

```
   JP_aplicacion.add(new JLabel("Videoclub versión 1.1"));
   JP_aplicacion.add(new JLabel("\n"));
   JP_aplicacion.add(new JLabel("Efren Zurita Alonso"));
   JP_aplicacion.add(new JLabel("\n"));
   JP_aplicacion.add(new JLabel("Proyecto fin de carrera Ingenieria tecnica de telecomunicación"));


   System.out.print("");

//Informacion del sistema

   JTextArea JTA_info=new JTextArea(5,55);
   JTA_info.setEditable(false);

   String pc ="Equipo: "+ System.getenv("COMPUTERNAME");
   String usuario="Usuario:  "+System.getProperty("user.name");
   String sistema ="Sistema Operativo: "+ System.getProperty("os.name");
   String procesador="Procesador: "+  System.getenv("PROCESSOR_IDENTIFIER");
   String java = "Version JDK: "+System.getProperty("java.version");
   String salida=usuario+"\n"+pc+"\n"+sistema+"\n"+procesador+"\n"+java;
   JTA_info.setText(salida);
   JTA_info.setFont(new Font("italic", Font.ITALIC, 14));
   JP_informacionEquipo.add(JTA_info);

      JButton JB_aceptar = new JButton("Salir");
   //Clase ActionListener para hacer que el botón cierre el JInternalFrame
       JB_aceptar.addActionListener(new ActionListener() {  //
    public void actionPerformed(ActionEvent e) {      //boton de salida de la aplicacion
    dispose(); //cierra solamente el JInternalFrame
```

Figura 3.7. Implementación Acerca De

Se puede comprobar con una sencilla llamada al método ***System.getenv ()*** se obtiene en un String el nombre de nuestro equipo y el procesador. Con la llamada al método ***System.getProperty ()*** se obtienen los datos del usuario, el sistema operativo y la versión del JDK.

A continuación todo ello se guarda en un String de salida que se incluye en una JTextArea para añadirse al panel principal de la ventana JP_informacionEquipo. Una vez finalizado este proceso, se han obtenido los datos y se plasman en la ventana que se ve al llamar al submenú Acerca de.

Finalmente se incluye el botón de salir al que se le añade la funcionalidad de salir de la ventana con el método *actionPerformed*(), como ya hemos explicado anteriormente, de la misma forma que el submenú salir nos servirá para salir de la aplicación

3.3.2 Actualizar

En este apartado se va a detallar el menú actualiza, el cual consta de varios submenús cada uno de los cuales con una función distinta. A continuación vamos a explicar cada uno de ellos.

- **Ingresar Clientes**: este submenú consta una ventana la cual sirve para añadir los datos del nuevo cliente que formara parte del videoclub, estos datos son: su nombre, teléfono, dirección, e-mail, DNI o pasaporte.

  Cada cliente tendrá asociado un número de identificación que aparecerá arriba de la ventana, de manera que conforme se vayan añadiendo más clientes el número de identificación será mayor.

- **Ingresar Película**: en este submenú aparece una ventana parecida a la del submenú ingresar clientes en la que se irán añadiendo los datos referentes a la nueva película que tendremos disponible para nuestro videoclub virtual, para ello también se utilizan radio botones para poder elegir la clasificación de la película, es decir, si la película está en estreno o no.

- **Borrar Cliente**: este submenú consta de una ventana pequeña en la cual aparece un espacio para que se indique el número de identificación del cliente que se quiera borrar de la base de datos del videoclub

- **Borrar Película**: en este submenú podemos ver una ventana que realiza la misma función que la ventana descrita en borrar cliente, pero esta vez para una película.

- **Actualizar Clientes**: al pulsar esta opción dentro del menú Actualizar se puede ver una ventana en la cual indicando el identificador de cliente que se quiera actualizar y mostrando los datos de ese cliente tendremos que ir modificando los datos de ese cliente menos el identificador para posteriormente poder actualizar los datos modificados.

- **Actualizar Película**: en este submenú podemos apreciar una ventana parecida a la de Actualizar Clientes en la cual, dependiendo del identificador de película que se introduzca se podrán ir actualizando los datos de la película que queramos cambiar, para finalmente poder actualizar los datos referidos a dicha película.

3.3.2.1 Ingresar

3.3.2.1.1 Ingresar Clientes

La Figura 3.8 muestra la ventana que aparece cuando dentro del menú Actualiza, se pulsa la opción ingresar clientes.

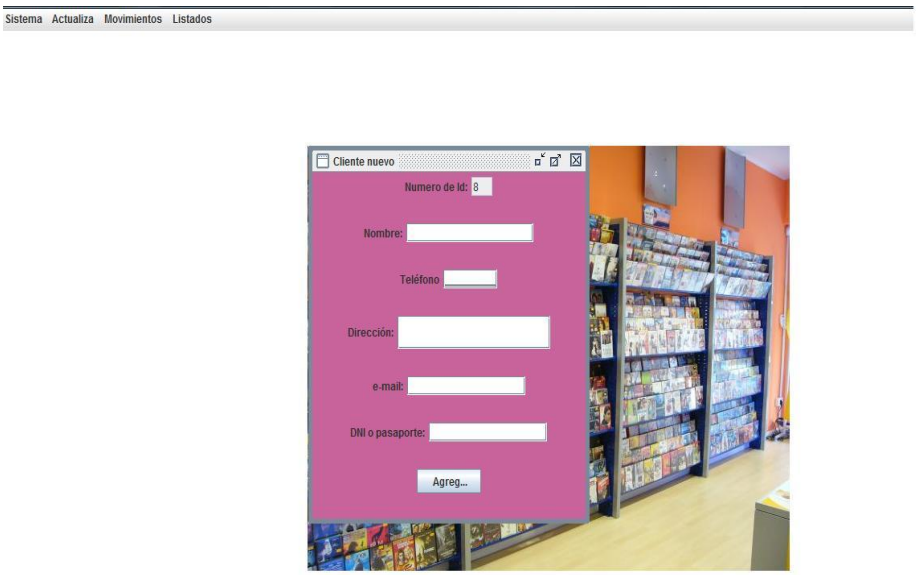


Figura 3.8. Ingresar Clientes

Al abrirse la ventana del submenú de ingresar clientes se puede apreciar un formulario donde habrá que ir rellenando para crear el cliente nuevo, los datos del cliente se introducen en etiquetas (JLabel) o áreas de texto (JTextArea). Una vez se introducen los datos del cliente para agregar finalmente el cliente a la base de datos del videoclub se pulsa el botón Agregar. La Figura 3.9 muestra se puede apreciar el código que hemos usado para la implementación del submenú insertar clientes.

```
JL_direccionCli = new JLabel("Dirección:");
JL_emailCli = new JLabel("e-mail:");
JL_dniPasaporteCli = new JLabel("DNI o pasaporte:");

//instanciamos los campos de texto

JTF_id = new JTextField(2);
JTF_nombreCli = new JTextField(14);
JTF_telefonoCli = new JTextField(15);
try{
   MaskFormatter formatter = new MaskFormatter("#########");
   formatter.setPlaceholderCharacter('_');
      JL_telefonoCli=new JLabel("Teléfono");
      JTF_telefonoCli = new JFormattedTextField(formatter);
}
catch(Exception er){
    System.out.println(er);
}//final del catch
JTF_emailCli = new JTextField(13);
JTF_dniPasaporteCli = new JTextField(13);

//instanciamos las areas de texto

JTA_direcCli = new JTextArea(2, 17);
JTA_direcCli.setWrapStyleWord(true);
JScrollPane JSP_direccionCliente= new JScrollPane(JTA_direcCli,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,JScrollPane.HORIZONTAL_SCROLLBAR_AS_NE

//instanciamos los botones
```

Figura 3.9. Implementación Insertar Clientes

El código de la implementación de este apartado consta de una serie de etiquetas:

- ✓ **JL_id**: etiqueta para el identificador de cliente
- ✓ **JL_nombreCli**: etiqueta para el nombre del cliente
- ✓ **JL_telefonoCli**: etiqueta para el teléfono del cliente
- ✓ **JL_direccionCli**: etiqueta para la dirección del cliente
- ✓ **JL_emailCli**: etiqueta para el email del cliente
- ✓ **JL_dniPasaporteCli**: etiqueta para el DNI o el pasaporte del cliente.

El código también incluye un campo de texto JTextField y un área de texto JTextArea, donde irán los datos referidos al cliente. Cada etiqueta o área de texto tiene un tamaño preciso.

Si bien dentro de las aplicaciones bajo swing (JAVA) podemos utilizar un componente como el JTextField para capturar una entrada de texto, esto no lo podemos hacer igual si esta entrada debe tener un formato específico o es necesario un tamaño obligatorio como por ejemplo a la hora de capturar un número de teléfono o un DNI. Por ello, utilizamos un componente conocido como JFormattedTextField, al cual, podemos indicar el formato de nuestra entrada. Lo que hacemos es crear una instancia de la Clase **MaskFormatter** en base a un patrón y este enviarlo en el constructor del JFormattedTextField.

MaskFormatter formatter = new MaskFormatter (“#########”);

JFormattedTextField JTF_TelefonoCli = new JFormattedTextField(formatter);

Hay que tener en cuenta el entorno de desarrollo NetBeans, que es el entorno que se utiliza para desarrollar la aplicación para el Proyecto Fin de Carrera, por lo que se coloca una instancia de la clase AbstractFormatterFactory en la propiedad FormatterFactory del elemento gráfico

El trozo de código que aparece en la figura 3.10 detalla cómo está construido el método actionPerformed () de la clase insertarClientes.java. Este código será muy similar al utilizado para las clases Borrar.java y Actualizar.java.

```
public void actionPerformed(ActionEvent e){

    boolean llenadoCorrecto = false;


 if(e.getSource()== JB_insertarCli){//si la fuente del action event es insertar cliente

       setId(JTF_id.getText());
       setNombre(JTF_nombreCli.getText());// el nombre que lo hemos puesto en el textfield lo establecemos.
       setTelefono(JTF_telefonoCli.getText());
       setDireccion(JTA_direcCli.getText());
       setEmail(JTF_emailCli.getText());
       setDniPasp(JTF_dniPasaporteCli.getText());

 //si el string recibido es 0 se pondrá un mensaje indicando el error
 if(   getDniPasp().length()==0||getTelefono().length()==0||getNombre().length()==0||getDireccion().length()==0||getEmail().length()==0){
 JOptionPane.showMessageDialog(null, "Rellena lo qe falta", "ERROR!", JOptionPane.WARNING_MESSAGE, imagen1IC);
 }

 else if(!JTF_nombreCli.getText().matches("[a-zA-Z]+\\s+[a-zA-z]+"))
     JOptionPane.showMessageDialog(null, "nombre invalido");
 else if(!JTF_telefonoCli.getText().matches("\\d{9}"))
     JOptionPane.showMessageDialog(this, "Formato de teléfono incorrecto");
 else if(!JTF_dniPasaporteCli.getText().matches("\\S{9}"))
     JOptionPane.showMessageDialog(null, " DNI o pasaporte\ndel cliente");


 else{
      llenadoCorrecto=true;
```

Figura 3.10. Action Performed

Con esto nuestra entrada será validada en este caso para números de nueve dígitos. También tenemos que tener en cuenta que si utilizamos el entorno de desarrollo NetBeans, deberemos colocar una instancia de la clase AbstractFormatterFactory en la propiedad FormatterFactory del elemento gráfico.

3.3.2.1.2Ingresar Película

La Figura 3.11 muestra la ventana que aparece cuando dentro del menú Actualiza, se pulsa la opción insertar película.

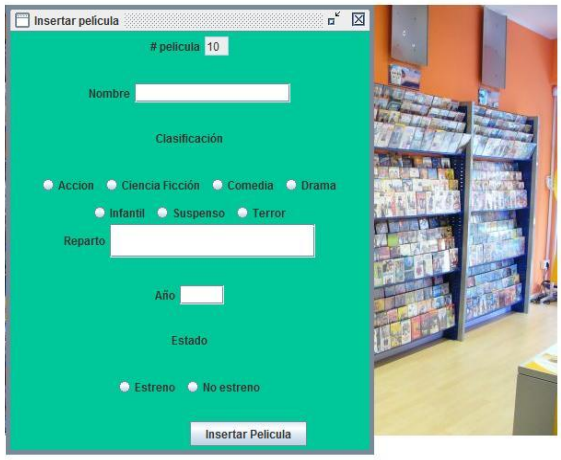


Figura 3.11. Insertar Película

Cuando el usuario accede a este submenú, aparece esta ventana en la cual, podemos apreciar tal y como se ve en la Figura 20 como se compone de una serie de etiquetas y áreas de texto para que el usuario vaya insertando los datos referentes a la película. La principal diferencia respecto a la ventana de ingresar clientes radica en el uso de radio botones. Los botones de la opción RadioButton se utilizan para seleccionar solo un elemento de un conjunto de elementos. No es lo mismo que un CheckBox, ya que un conjunto de RadioButton solo puedes seleccionar un elemento mientras que en un conjunto de CheckBox puedes seleccionar uno, varios o todos los elementos. Swing soporta radio botones con las clases JRadioButton y ButtonGroup. Se puede apreciar en la Figura 3.12 como se crean los radio botones.

```
//instanciamos los RadioButton

JRB_accion = new JRadioButton("Accion");
JRB_cienciaFiccion = new JRadioButton("Ciencia Ficción");
JRB_comedia = new JRadioButton("Comedia");
JRB_drama = new JRadioButton("Drama");
JRB_infantil = new JRadioButton("Infantil");
JRB_suspenso = new JRadioButton("Suspenso");
JRB_terror = new JRadioButton("Terror");

JRB_noEstreno = new JRadioButton("No estreno");
JRB_estreno = new JRadioButton("Estreno");

// instanciamos los ButtonGroup

BG_Clasif = new ButtonGroup();
BG_estado = new ButtonGroup();
```

Figura 3.12 Implementación Insertar Película

En la Figura 3.13 se muestra un trozo de código en el que se trata un tipo de evento especial para los RadioButton. En vez de asignar un ActionListener a cada botón, lo que se hace es asignar un ItemListener a nuestros radio botones de manera que se pueda elegir uno de los radio botones que se nos muestren en la clasificación de la película o en el estado de la película.

```
JRB_infantil.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            setClasificacion("infantil");
        System.out.println(getClasificacion());
        }
    });


JRB_infantil.setBackground(new java.awt.Color(0,200,155));

// Clase interna del JRB_suspenso, la clasificacion de la pelicula pasa a ser de "suspense"

JRB_suspenso.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            setClasificacion("suspense");
        System.out.println(getClasificacion());
        }
    });

JRB_suspenso.setBackground(new java.awt.Color(0,200,155));

// Clase interna del JRB_terror, la clasificacion de la pelicula pasa a ser de "terror"

JRB_terror.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            setClasificacion("terror");
        System.out.println(getClasificacion());
        }
    });
```

Figura 3.13.ItemStateChanged

Como se puede observar en la imagen anterior sobrescribimos el método *itemStateChanged (ItemEvent e){}* de manera que cada vez que se pulse sobre ese radio botón, establezcamos la clasificación de la película, por ejemplo, para el radio botón *JRB_infantil* estableceremos la clasificación de la película "*infantil*" cada vez que el botón haya sido seleccionado. Todo esto lo haremos con cada uno de los radio botones al que les añadiremos su escuchador correspondiente y trataremos el evento que le corresponde.

En la Figura 3.14 se puede apreciar el actionPerformed () correspondiente al botón de insertar película que es muy similar al de la clase Insertar Cliente, ya que el botón realiza la misma función.

```
public void actionPerformed(ActionEvent e){

        boolean llenadoCorrecto = false;

        try{
    if(e.getSource()== JB_insertar){//si la fuente del action event es insertar pelicula
    //obtenemos los datos de los campos de texto para que sean procesados..

          setId(JTF_numPelicula.getText());
          setNombre(JTF_nombrePelicula.getText());
          setAño(JTF_ano.getText());
          setReparto(JTA_reparto.getText());

    //si el string recibido es 0 se pondrá un mensaje indicando el error
    if(getNombre().length()==0||getAño().length()==0||getReparto().length()==0||(getEstado().equals("false")==true)||(getClasificacion().equal
    JOptionPane.showMessageDialog(null, "Complete lo que falta", "ERROR!", JOptionPane.WARNING_MESSAGE);
    }
    else if(!JTF_ano.getText().matches("\\d{4}"))
        JOptionPane.showMessageDialog(null, "Formato de año incorrecto", "ERROR", JOptionPane.WARNING_MESSAGE);

    else{
          escribeArchivo();
          escribeID();
          JTF_ano.setText("");
          JTF_nombrePelicula.setText("");
          JTA_reparto.setText("");
          JTF_numPelicula.setText(""+UltimoID());//coge la ultima pregunta y le suma 1 para obtener el nuevo id
```

Figura 3.14.Inplementacion Action Performed Insertar Película

Una vez hemos rellenado bien los datos referentes a la nueva película que queremos añadir a la base de datos de nuestro videoclub virtual, si los formatos de año son correctos y hemos rellenado correctamente los datos, llamaremos a los métodos ***escribeArchivo*** *()* y ***escribeID*** *()*. A continuación al llamar al método ***escribeArchivo****() se crea* un fichero llamado "Películas.txt"

en el cual estarán todos los datos referidos a esas películas. Posteriormente al llamar al otro método ***escribeID ()*** se crea otro fichero llamado "ID_pelicula.txt" en el que estarán todos los números de identificador de las películas que tenemos en la base de datos.

Para leer y escribir, respectivamente en un fichero, usaremos las clases **FileReader** y **FileWriter**. Para escribir, se usa el método write de **FileWriter**, este método usa como parámetro un String o un número que corresponda a un carácter de la tabla ASCII.
Para leer, usaremos el método read de **FileReader**, este método no tiene parámetros pero devuelve un número, al cual, habrá que hacerle un casting para pasarlo a Char. Cuando termina el fichero el método read devuelve -1.

3.3.2.2Borrar

3.3.2.2.1Borrar Clientes

La Figura 3.15 muestra la ventana que aparece cuando dentro del menú Actualiza, se pulsa la opción Borrar Cliente.

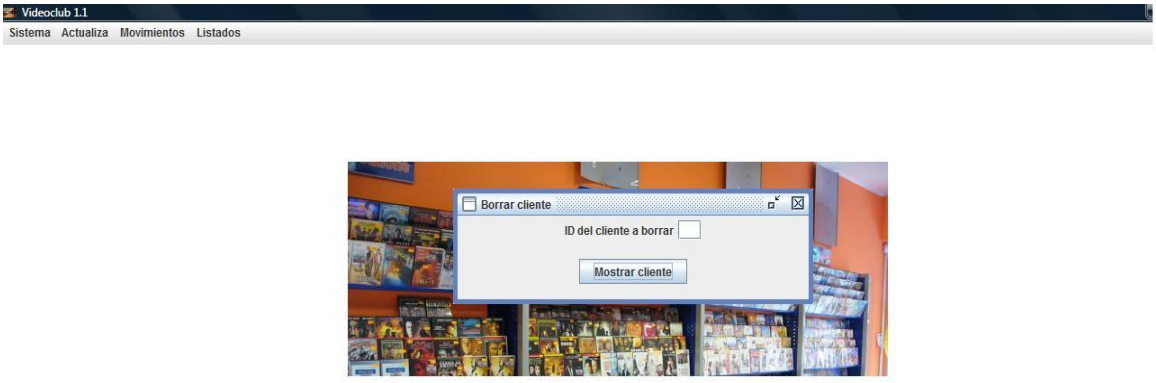


Figura 3.15.Borrar Clientes

El usuario introducirá el identificador del cliente que se quiere borrar y pulsara sobre el botón "Mostrar Cliente". A continuación, aparecen los datos del cliente que el usuario quiere borrar de la base de datos de la aplicación, el usuario aceptará la operación y el cliente desaparecerá definitivamente de la base de datos. Como se puede apreciar hay una etiqueta con un pequeño

campo de texto para introducir el identificador del cliente, y un botón por lo que el código de la interfaz gráfica es bastante sencillo para este apartado.

Como se aprecia en la Figura 3.16, en el método actionPerformed() se lleva a cabo la funcionalidad del botón para que borre al cliente en cuestión. Una vez introducido bien el número de identificador de cliente y se verifica que corresponde con un cliente de la base de datos del videoclub se muestra un JOptionPane, un panel en el que se nos preguntara si queremos borrar el cliente de nuestra aplicación. A continuación si pulsamos aceptar se llama al método eliminarCliente(identificador) que será el encargado de borrar el cliente de nuestra base de datos.

```
public void actionPerformed(ActionEvent e){

    boolean datoCorrecto=false; //variable que muestra si el cliente esta o no dentro del archivo de texto

    identificador= JTF_identific.getText();

    if(identificador.length()==0)
        JOptionPane.showMessageDialog(null, "Pon el identificador", "ERROR", JOptionPane.ERROR_MESSAGE);
    else{
        if(buscarCliente(identificador)==true)
         datoCorrecto=true;
        else{
            JOptionPane.showMessageDialog(null, "¡El cliente no ha sido encontrado!", "ERROR", JOptionPane.WARNING_MESSAGE);
            JTF_identific.setText("");
        }

      if(datoCorrecto==true){
      JOptionPane.showMessageDialog(null, datosDelClienteParaBorrar,"DATOS DEL CLIENTE "+identificador, JOptionPane.INFORMATION_MESSAGE);
      int borrar= JOptionPane.showConfirmDialog(null, "¿Quieres borrar el cliente?", "BORRAR CLIENTE", JOptionPane.YES_NO_CANCEL_OPTION);
      if(borrar==0){
      eliminarCliente(identificador);
      JOptionPane.showMessageDialog(null, "El cliente se ha borrado");
      dispose();
      }
      else if(borrar==1){
         JOptionPane.showMessageDialog(null, "No ha borrado el cliente " + JTF_identific.getText());
         JTF_identific.setText("");
      }//fin else
      else if(borrar==2){
         //JOptionPane.showMessageDialog(null, "", "CANCELADO", JOptionPane.INFORMATION_MESSAGE);
         dispose();
```

Figura 3.16.Action Performed Borrar Clientes

En la Figura 3.17 y en la Figura 3.18 se puede apreciar el código detallado del método eliminarCliente(identificador) llamado desde el actionPerformed() que elimina un cliente de la base de datos de nuestra aplicación.

```java
/* metodo eliminarCliente con el que borramos un cliente de la lista de clientes, usamos un archivo temporal para pegar los
   clientes del clientes.txt al temporal.txt menos el cliente con el id que le pasamos*/
 public void eliminarCliente(String Idcliente){
    //archivo clientes.txt donde encontramos la informacion de los clientes
    File archivo=new File("Clientes.txt");
    //archivo temporal.txt  archivo temporal con el que realizamos la actualizacion de la informacion
    File temporal=new File("temporal.txt");

     try{
      // abrimos el archivo de lectura clientes.txt
      FileReader archivoLectura=new FileReader(archivo);
      BufferedReader buffer=new BufferedReader(archivoLectura);
      //abrimos el archivo de escritura temporal.txt para pasar los registros desde el archivo clientes.txt
      FileWriter archivoEscritura=new FileWriter(temporal);
      BufferedWriter bufferTemp=new BufferedWriter(archivoEscritura);
      String linea=buffer.readLine();

      // recorremos los registros del archivo principal clientes.txt
      while(linea!=null){//mientras la linea del archivo no esté vacia se seguirá leyendo el archivo

         String[] registro=linea.split(";");
         //si el registro tiene el mismo id del cliente a eliminar no se agrega al nuevo archivo, ya que si es el mismo es el id que le hemos pasado
         //sobre el cliente que queremos borrar, y en el buffer de salida de temporal.txt no queremos guardar ese cliente que va a ser borrado.
         if(!registro[0].equals(Idcliente)){
           bufferTemp.write(linea);
           bufferTemp.newLine();

         }

         linea=buffer.readLine();
```

Figura 3.17.Implementación EliminarCliente

Como se aprecia este método sirve para borrar un cliente de la lista de clientes, para ello usamos un archivo temporal llamado “temporal.txt” en el que pegamos todos los clientes del archivo Clientes.txt salvo el cliente que queremos eliminar, es decir, si el registro tiene el mismo identificador del cliente que queremos eliminar no se agrega al nuevo archivo, de esta manera en el nuevo archivo tendremos una nueva lista de clientes que cuenta con todos los clientes que teníamos anteriormente menos el que se ha borrado.

```
            linea=buffer.readLine();
        }//final del del while
         //cerramos temporal.txt y clientes.txt
        buffer.close();
        bufferTemp.close();

                try {
                        File inFile = new File("temporal.txt");
                        File outFile = new File("clientes.txt");

                        FileInputStream in = new FileInputStream(inFile);
                        FileOutputStream out = new FileOutputStream(outFile);

                        int c;
                        while( (c = in.read() ) != -1)//pasamos los datos que queden por leer del temporal.txt al clientes txt
                                out.write(c);

                        in.close();
                        out.close();
                } catch(IOException ioE) {
                        JOptionPane.showMessageDialog(this,"Hubo un error a la hora de borrar");
                }

        }//final del try

        catch(Exception ex){
         ex.printStackTrace();

        }
    }//final del metodo eliminar cliente
```

Figura 3.18.Implementación EliminarCliente2

A continuación, una vez obtenido el archivo “temporal.txt” que contiene la lista de clientes modificada después de que el usuario borrase el cliente que el usuario ha decidido borrar se vuelca el contenido de este archivo en el archivo “clientes.txt” (por eso el nombre de temporal, ya que se usa como puente hacia este archivo), que será el archivo definitivo en el que tendrá todos los clientes que formen parte del videoclub una vez realizado el borrado del cliente.

3.3.2.2.2 Borrar Película

En la Figura 3.19 se muestra la ventana que aparece cuando dentro del menú Actualiza, se pulsa la opción Borrar Película.

Figura 3.19.Borrar Película

Como se puede observar en la imagen tenemos una ventana muy similar a la ventana que aparece cuando un usuario quiere borrar un cliente. Cuando el usuario desee borrar una película determinada de la base de datos del videoclub tiene que indicar el ID de la película que quiere eliminar para a continuación pulsar el botón de mostrar película, una vez esto, aparecerán los datos de la película y la aplicación preguntara al usuario si desea finalmente borrarlo de la base de datos. El usuario pulsará aceptar si así lo desea. Por otra parte se puede ver como el código de la clase borrarPelicula.java es muy similar al código de la clase borrarClientes.java.

En el método actionPerformed() se va a llamar también a dos métodos eliminarPelicula(identificador) y buscarPelícula(identificador) que es un método similar al buscarCliente(identificador) que teníamos en la clase borrarClientes.java.

En las Figuras 3.20 y 3.21 se muestra el método buscar película. Este método comprueba que la película que estamos buscando se encuentra dentro de la base de datos de nuestro videoclub virtual. Para ello, únicamente el usuario le pasara el identificador de la película y de esta manera se procederá a buscar la película y eliminarla de la base de datos.

```
public boolean buscarPelicula(String identificador){

insertPeli = new insertarPelicula(null, null, null, null, null, null);

boolean encontrado= false;

BufferedReader entradaDeArchivo = insertPeli.getBufferedReader();

try{
    String registroActual = entradaDeArchivo.readLine(); //obtenemos la linea del registro.
    while(registroActual!= null && encontrado==false){

        StringTokenizer st = new StringTokenizer(registroActual, ";");

        String id="", nombre="", clasificacion="", reparto="", año="", estado="";

        int ID=1, NOMBRE=2, CLASIFICACION=3, REPARTO=4, AÑO=5, ESTADO=6;


        int contador=1; //cuando contador esté en algún valor de los anteriores, se guarda ese String en la variable correspondiente

        while(st.hasMoreTokens()){ //sigue en el ciclo mientras hallan mas tokens
            if(contador==ID)
                id=st.nextToken();
            else if (contador==NOMBRE)
                nombre=st.nextToken();
            else if(contador==CLASIFICACION)
                clasificacion=st.nextToken();
            else if(contador==REPARTO)
                reparto=st.nextToken();
```

Figura 3.20.Buscar Película

```
            else if(contador==AÑO)
                año=st.nextToken();
            else if(contador==ESTADO)
                estado=st.nextToken();
            contador++;
        }//final del while
        if(identificador.equalsIgnoreCase(id)){
            encontrado = true;
            //datosPeliculaBorrar es la variable con la que guardamos los datos de la pelicula que queremos borrar y los muestra mediante un mens
            datosPeliculaBorrar = "ID"+id+"\nNombre: "+nombre+"\nClasificación: "+clasificacion+"\nReparto: "+reparto+"\nAño: "+año+"\nEstado: "+
        }//final del if
        else
            registroActual= entradaDeArchivo.readLine();
    }//final del while
}//final del try
catch(FileNotFoundException fnfe){
JOptionPane.showMessageDialog(null, fnfe);
}//final del catch
catch(IOException ioE){
JOptionPane.showMessageDialog(null, ioE);
}//final del catch

return encontrado;
}//final del metodo buscarPelicula
```

Figura 3.21.Buscar Película II

Analizando con más detalle este método, se aprecia cómo se usa la clase **StringTokenizer[**3.3.1**]**. Esta clase es usada para dividir una cadena de texto en Tokens y extraer el contenido que nos interesa. Un token es la secuencia máxima de caracteres consecutivos que no son delimitadores.

Como se ha comentado anteriormente, el identificador de la película que el usuario le ha pasado como argumento coincide con alguno de los identificadores de las películas que hay en el

archivo, se encuentra la película que se buscaba. En caso contrario, se siguen leyendo las líneas del archivo hasta encontrarlo.

3.3.2.3Actualizar

Desde este menú el usuario podrá realizar cambios en todo lo referido a los datos de las películas o clientes que se encuentren en la base de datos del videoclub virtual. Para un cliente se podrá actualizar su nombre, su dirección, su email, su teléfono y su DNI o pasaporte, ya que en algún momento los datos previos podían no ser correctos. Para una película los datos que se podrán modificar son su nombre, su clasificación (el tipo de película que es, ya sea comedia, suspense, infantil, acción, ciencia ficción, terror o drama), el reparto, el año de estreno y su categoría (si es de estreno o no).

3.3.2.3.1Actualizar Clientes

Si el usuario quiere modificar algún dato referido al cliente, ya sea porque ha cambiado su número de teléfono o su domicilio particular, o bien no se introdujo anteriormente un dato de forma correcta, en este apartado de actualizar clientes el usuario lo podrá hacer desde esta ventana. A continuación en la Figura 3.22, se puede apreciar la ventana que le aparecerá al usuario.

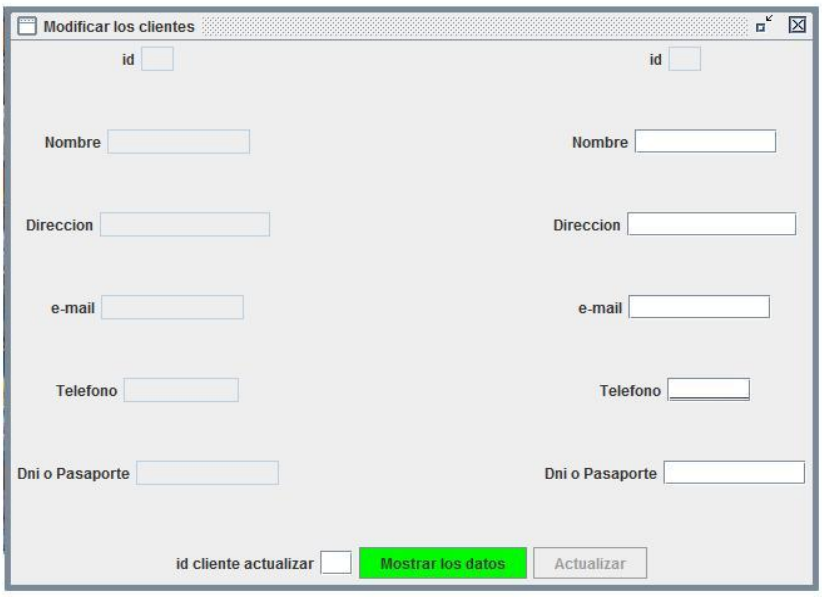


Figura 3.22.Actualizar Clientes

Como se puede apreciar esta ventana es más grande debido a la necesidad de ver en una misma ventana los datos que el usuario necesitara modificar del cliente y los nuevos datos para ese cliente después de actualizarlos.

Para actualizar los datos, el usuario tendrá que poner en la parte inferior el identificador del cliente sobre el que quiere actualizar los datos y pulsar el botón de mostrar los datos, en la parte derecha podrá modificar los datos y cuando esté listo pulsar el botón actualizar. Como podemos comprobar el panel de actualizar clientes consta de varias etiquetas y áreas de texto además de los botones en la parte de debajo de mostrar los datos y actualizar.

La interfaz grafica es sencilla y cómoda para que el usuario no encuentre problema alguno. En las Figuras 3.23, 3.24 y 3.25 se muestra el código implementado para la clase ActualizarClientes

```
if(e.getSource()==JB_actualizar){//si la fuente del action event es actualizar los datos

    int seleccion = JOptionPane.showOptionDialog( JB_actualizar,"¿Has actualizado bien los datos?","Cancelar",JOptionPane.YES_NO_CANCEL_OPTION,JOption
    if(seleccion==0){
    //Se obtienen los datos de los campos de texto para procesarlos.
    setNuevoNom(JTF_nombreClimodif.getText());
    setNuevaDirec(JTF_direccionClimodif.getText());
    setNuevoemail(JTF_emailClimodif.getText());
    setNuevoTelef(JTF_telefonoClimodif.getText());
    setNuevoDniPasp(JTF_dniPasaporteClimodif.getText());
    //escribeNuevoCliente(identificador);
    //si algún dato no lo hemos ingresado o algun campo este vacio o si el tamaño del String  es cero, desplegaríamos un mensaje indicando el error
    if(getNuevoNom().length()==0 || getNuevaDirec().length()==0 || getNuevoemail().length()==0 || getNuevoTelef().length()==0 || getNuevoDniPasp().len
        JOptionPane.showMessageDialog(null, "rellena todos los campos");

    else if(!JTF_nombreClimodif.getText().matches("[a-zA-Z]+\\s+[a-zA-z]+"))
        JOptionPane.showMessageDialog(null, " nombre y apellido");
    else if(!JTF_telefonoClimodif.getText().matches("\\d{9}"))
        JOptionPane.showMessageDialog(this, "Formato de teléfono incorrecto");
    else if(!JTF_dniPasaporteClimodif.getText().matches("\\S{9}"))
        JOptionPane.showMessageDialog(null, "DNI o pasaporte\ndel cliente");

    else{
        actualizarCliente(getIdentificador(), getNuevoNom(), getNuevaDirec(), getNuevoemail(), getNuevoTelef(), getNuevoDniPasp());
        dispose(); //si cumplimos los requisitos y no nos sale ningun mensaje de error como los 4 anteriores actualizamos y la ventana se cierra
    }//final del else
}
```

Figura 3.23 Implementación Actualizar Clientes

Como se observa, en el método actionPerformed (ActionEvent e) de la clase Actualizar Clientes que se aprecia en la figura de arriba y nos fijamos en la función que realiza el método cuando pulsamos el botón de actualizar los clientes, establecemos el nuevo nombre, la nueva dirección, el nuevo email, el nuevo teléfono y el nuevo DNI o pasaporte que el usuario a puesto en la JTextArea o área de texto y actualizamos el cliente con esos nuevos datos, siempre y

cuando los datos sean correctos, ya que como podemos comprobar los campos tienen que estar todos rellenos, no puede quedar ni uno vacío, el nombre y los apellidos tienen que ser correctos, así como el formato de teléfono que consta de 9 dígitos numéricos y el formato del DNI o pasaporte.

A continuación vamos a ver un trozo de código del método actualizarCliente que actualizara los datos del cliente en función de los datos pasados por parámetro. En las Figuras 3.24 y 3.25 se puede apreciar dicho código.

```
/* metodo actualizarCliente en el que creamos dos archivos uno de lectural donde vamos a leer los datos y otro de escritura donde los vamos a escri
 public void actualizarCliente(String idcliente, String nombre, String direccion, String email, String telefono, String autorizado){

     File archivo=new File("clientes.txt");

     File temporal=new File("temporal.txt");//archivo temporal en el que realizamos la actualizacion de la informacion

      try{

       FileReader archivoLectura=new FileReader(archivo);// abrimos el archivo antes de leerlo
       BufferedReader buffer=new BufferedReader(archivoLectura);

       FileWriter archivoEscritura=new FileWriter(temporal);//abrimos el archivo temporal de escritura para pasar los registros desde el archivo cl
       BufferedWriter bufferTemporal=new BufferedWriter(archivoEscritura);
       String linea=buffer.readLine();//leemos una linea del archivo clientes.txt

       // recorremos los registros del archivo principal clientes.txt
       while(linea!=null){

          String[] registro=linea.split(";");//se toman los datos hasta -

          if(!registro[0].equals(idcliente)){//si registro[0] es diferente al id del cliente
            bufferTemporal.write(linea);//escribimos en el archivos los datos de los otros clientes
            bufferTemporal.newLine();//se crea una nueva linea


          }
```

Figura 3.24 Implementación Actualizar Clientes2

```
    else if(registro[0].equals(idcliente)){
        bufferTemporal.write(idcliente+";"+nombre+";"+direccion+";"+email+";"+telefono + ";" + autorizado+"\n");//escribimos en el archivo los n
    }


    linea=buffer.readLine();//leemos la siguiente linea del archivo
}
 //cerramos los archivos clientes.txt y temporal.txt
buffer.close();//cerramos el BufferReader
bufferTemporal.close();//cerramos el BufferWriter

        try {
                File inFile = new File("temporal.txt");
                File outFile = new File("clientes.txt");

                FileInputStream in = new FileInputStream(inFile);//creamos el FileInputStream pasandole un archivo
                FileOutputStream out = new FileOutputStream(outFile);//creamos el FileOutputStream pasandole un archivo

                int c;
                while( (c = in.read() ) != -1)//mientras que existan datos en el archivo temporal.txt
                        out.write(c);//pasamos lo que tenemos de temporal.txt a clientes.txt

                in.close();//se cierra el FileInputStream
                out.close();//se cierra el FileOutputStream
        } catch(IOException e) {
                JOptionPane.showMessageDialog(this,"Error al actualizar");
        }//final del catch

}//final del try
```

Figura 3.25 Implementación Actualizar Clientes3

Este método es muy parecido al método borrarClientes, se crean dos ficheros, uno de lectura para leer los datos y otros de escritura para escribirlos. Se usa un archivo temporal llamado "temporal.txt" que contendrá los clientes que había en el archivo "clientes.txt" pero con los datos actualizados.

A continuación una vez que tenemos el archivo "temporal.txt" con la lista de clientes modificada, se vuelca el contenido de este archivo en el archivo "clientes.txt", el cual, será el archivo definitivo, mediante el cual el usuario podrá consultar los datos ya modificados de todos los clientes que formen parte del videoclub.

En el constructor de la clase actualizarClientes se utiliza la clase MaskFormatted [3.3.2] que es usada para formatear y editar String.

3.3.2.3.2Actualizar Película

En la Figura 3.26 se muestra la ventana que le aparece al usuario cuando selecciona el submenú de Actualizar Película. Como se puede apreciar, la ventana tiene el mismo tamaño que la ventana que aparece cuando actualizamos un cliente y es bastante similar.

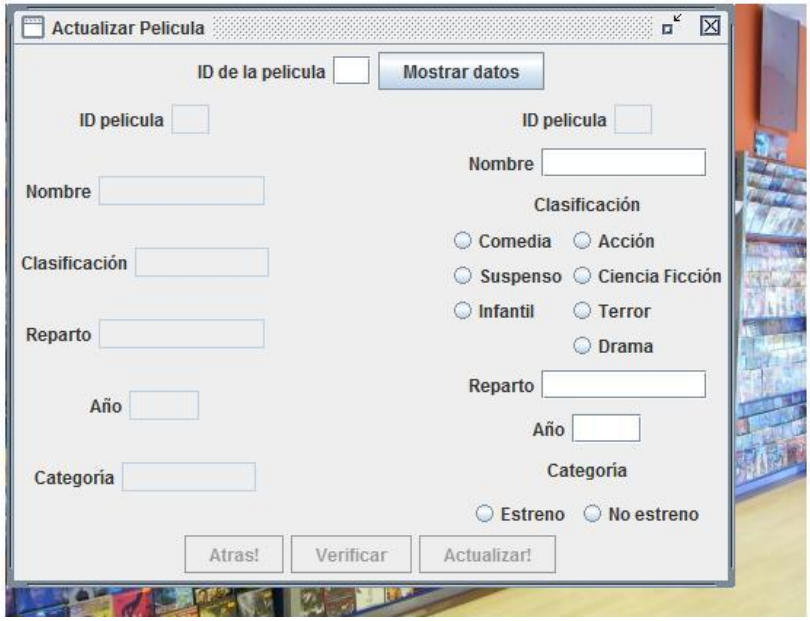


Figura 3.26.Actualizar Película

La principal diferencia entre esta ventana y la ventana de Actualizar clientes la podemos comprobar en la aparición de radio botones que usamos para elegir la clasificación de la película y su categoría, por lo demás es muy similar con la aparición de etiquetas, áreas de texto y cuatro botones.

Una vez abierta la ventana, cuando el usuario quiera modificar los datos de la película, tendrá que poner el identificador de la película en el área de texto de arriba y pulsar el botón de mostrar los datos.

Una vez se hallan rellenado todos los campos correctamente habrá que pulsar el botón verificar que comprobara que los datos están bien y posteriormente el botón actualizar con el que actualizaremos los datos finalmente.

En la Figura 3.27 se muestra el método actionPerformed (ActionEvent e) de la clase actualizarPelicula.java. El botón verificar, ya que no está implementado en la clase actualizarCliente.java, simplemente lo que hace es comprobar que los datos que hemos introducido son correctos, una vez los datos sean correctos dejamos que pulse el botón Actualizar para hacer la actualización de los datos. El año de estreno de la película debe estar comprendido entre 1900 y 2014 y ningún campo debe estar vacío. Una vez pulsado el botón de verificar ya no se podrá modificar nada salvo si pulsamos el botón de volver a atrás.

```
if(e.getSource()==JB_verificar){//si la fuente del action event es verificar los datos
try{
       setNuevoNombre(JTF_nombre2.getText());
       setNuevoReparto(JTF_reparto2.getText());
       setNuevoAño(JTF_año2.getText());

  if( getNuevaClasificacion().equals("false")||getNuevoEstado().equals("false")||getNuevoNombre().length()==0||getNuevoReparto().length()==0||getNue
      JOptionPane.showMessageDialog(null, "Rellena todos los campos");
      JB_actualizar.setEnabled(false);
 }//final del if
  else if(!JTF_año2.getText().matches("\\d{4}")||Integer.parseInt(JTF_año2.getText())<1900||Integer.parseInt(JTF_año2.getText())>2014)
     JOptionPane.showMessageDialog(null, "Año incorrecto\nPon un año comprendido entre 1900 y 2014", "ERROR", JOptionPane.WARNING_MESSAGE);

  else{
    datosCorrectos=true;
    JTF_nombre2.setEditable(false);
    JTF_año2.setEditable(false);
    JTF_reparto2.setEditable(false);
    JB_verificar.setEnabled(false);
    JB_atras.setEnabled(true);
  }//final del else
if(datosCorrectos==true){
   JB_actualizar.setEnabled(true);// habilitamos el boton de actualizar
}//final del if
}//final del try
catch(NullPointerException npe){
JOptionPane.showMessageDialog(null, "¡Rellena todos los campos!");
}
}//final del getSource JB_verifcar
```

Figura 3.27.Implementación Actualizar Película

3.3.3 Movimientos

En este apartado se va a detallar el menú Movimientos, el cual consta de varios submenús cada uno de los cuales con una función distinta. Este menú se compondrá de cuatro submenús que serán:

- **Alquiler de una película:**

  El usuario introduce el número de identificación del cliente para escoger la película que desee alquilar. La aplicación posteriormente nos informa de la fecha de entrega y la cantidad a pagar por el alquiler. Una vez finalizada la elección del usuario, si todo esta correcto, se realiza el alquiler.

- **Cierre mensual de alquileres:**

  Este apartado ofrece un informe de todos los movimientos realizados en cada mes y el total de ingresos que se ha obtenido ese mes a través de los alquileres de las películas. Para obtener este informe, el usuario tendrá que indicarle el mes a través de una lista desplegable y se genera el informe.

- **Recomendación de películas:**

  Este submenú ofrece al usuario una serie de películas en base a unos criterios personales: sexo, edad, género que más le guste. El usuario rellena un cuestionario y en base a este, se genera un informe con las películas más apropiadas para ese usuario.

- **Ver Cartelera:**

  Este submenú ofrece al usuario la posibilidad de ver la Cartelera que hay actualmente en los cines.

3.3.3.1Alquiler de una película

En la Figura 3.28 se muestra la ventana que aparece cuando el usuario dentro del menú movimientos selecciona el apartado de Alquiler de película. En esta ventana el usuario realizará el alquiler de la película.

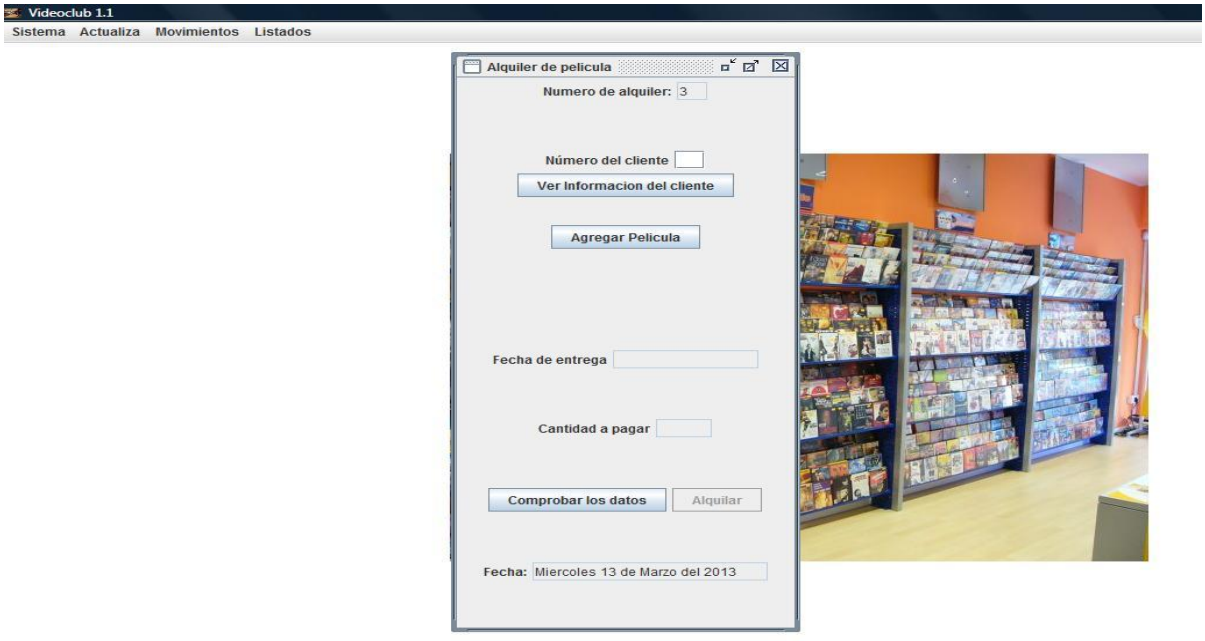


Figura 3.28.Alquiler de una película

Como podemos apreciar en la figura de arriba, esta interfaz gráfica está compuesta por varias etiquetas y campos de texto y varios botones. Lo primero que se puede apreciar es el número de alquiler, que ira avanzando a medida que se realicen más alquileres. Para poder alquilar una película, en primer caso habrá que poner el número de identificación del cliente.

A continuación tenemos el botón ver información del cliente que muestra un pequeño panel con los datos del cliente. El botón agregar película es donde el usuario realiza el alquiler. Al pulsar sobre este botón nos aparecerá una segunda ventana tal como se aprecia en la Figura 3.29.

Figura 3.29.Elegir película

Esta ventana será donde el usuario seleccione la película que quiera añadir a su alquiler, como se puede apreciar en la figura de arriba, indicándole el identificador de película correspondiente a la película que el usuario elija y pulsando el botón de buscar película, aparecerá la película siempre y cuando esa película forme parte de nuestra base de datos del videoclub.

A continuación, pulsando el botón agregar nos aparece las películas en el área de texto que tenemos, un área lo suficientemente grande ya que podemos incluir en nuestro alquiler varias películas si lo deseamos, en la parte de abajo podemos observar cómo se puede ver el precio total resultante de alquilar una o varias películas.

Una vez agregada la película en nuestro panel de alquiler de película nos aparecerá la fecha de entrega y la cantidad a pagar así como la fecha actual, por lo que tendremos ya todos los datos necesarios para realizar el alquiler si finalmente lo deseamos así.

En la clase alquilerDePelicula se observa como el método actionPerformed (ActionEvent e) desarrolla la funcionalidad para los botones al igual que lo hemos hecho en otras clases.

Á continuación pulsando el botón comprobar los datos se observa cómo se hace una llamada a los métodos validarCliente() y establecePeliculasPrecio() que se explican en las figuras 39 y 40.

En la Figura 3.30 se muestra el método **validarCliente**() para la clase alquilerDePelicula. Este método nos dice si el cliente forma parte de la base de datos de la aplicación.

```
/*Clase que valida el cliente si el id indicado del cliente se encuentra en el archivo*/

        public void validarCliente(){
            insertCli = new insertarClientes(null, null, null, null, null, null);
            if(buscarCliente(JTF_numeroCliente.getText())==true){


                System.out.println("Cliente encontrado");
                }//final del del if
            else{
                JOptionPane.showMessageDialog(null, "El cliente no ha sido encontrado", "¡ERROR!", JOptionPane.WARNING_MESSAGE);
                JTF_numeroCliente.setText("");
            }//final del else
        }//final del metodo validarPelicula
```

Figura 3.30.Validar Cliente

Una vez llamado al método validarCliente () y se comprueba que el cliente existe se llama al método buscarCliente (String identificador) pasándole como argumento un identificador de cliente, método que ya hemos explicado anteriormente en otros apartados.

En la Figura 3.31 se muestra el método **establecePelículasPrecio**() para la clase alquilerDePelicula. Este método establece el precio a pagar en función del número de alquileres que se hayan hecho. Dicho precio, se muestra en el panel alquiler de película a continuación de la etiqueta total a pagar. Para ello se ha creado un nuevo archivo llamado SeleccionPeliculas.txt en el que aparece por un lado el precio total a pagar y por otro las películas que se alquilen.

```
public void establecePeliculasPrecio(){

if(totalAPagar.equalsIgnoreCase("nada")==false){
    File f2 = new File("seleccionPeliculas.txt");
    try{
    FileInputStream fis = new FileInputStream(f2);
     InputStreamReader isr = new InputStreamReader(fis);
     BufferedReader br = new BufferedReader(isr);
     String datos = br.readLine();
     int i = datos.length();
     datos=datos.substring(0,i-2);
     System.out.println(datos);
     StringTokenizer st = new StringTokenizer(datos,";");
     totalAPagar=st.nextToken();
     peliculas=st.nextToken();
     JTF_cantidadPagar.setText(totalAPagar);
     setPeliculas(peliculas);
     f2.delete();

    }//final del try
    catch (FileNotFoundException fne){
        JOptionPane.showMessageDialog(null,"Debe de elegir una pelicula por lo menos");
    }//final del catch
       catch(IOException y){
           JOptionPane.showMessageDialog(null,"No se puede acceder al archivo con los datos de la pelicula");
       }//final del catch
}//final del if

else{
JOptionPane.showMessageDialog(null,"Añade al menos una pelicula");
```

Figura 3.31.Establecer precio de las películas

Para finalizar este apartado se explica más detalladamente la clase Seleccionpelicula.java, cuya clase tiene la funcionalidad para seleccionar la película que el usuario quiera alquilar.

En las Figuras 3.32 y 3.33 se puede apreciar el código que se usa en el método actionPerformed () para la clase Seleccionpelicula.java.

```
public void actionPerformed(ActionEvent e) {
if(e.getSource() == JB_buscarPeli){ //  la fuente del acton event es buscar pelicula
    int id=0;
    try{
    id=Integer.parseInt(JTF_numerocam.getText()); // coge el id de lo que pones en el text field
      if(validarPelicula(""+id)==true){// si el id es correcto(la pelicula existe)
            JTF_infoPeli.setText(meterPeli.getNombre());// incluye en el text field el nombre de la película
            JB_agregar.setEnabled(true);
       }
    }
    catch(NumberFormatException nfe){
       JOptionPane.showMessageDialog(null,"Debe poner un numero valido\n"+nfe);

    }//final del catch
      }

       if(e.getSource()== JB_agregar){// si la fuente del action event es agregar

       /* en la text area vamos poniendo los nombres de las películas mientras antes usamos el string peliculasJTA*/
      peliculasJTA=peliculasJTA+meterPeli.getNombre()+"  "+ meterPeli.getAño()+" "+meterPeli.getClasificacion()+" "+meterPeli.getEstado()+"\n";
      JTA_todasLasPelis.setText(peliculasJTA); //en la text area vamos poniendo los nombres de las películas su año clasificacion y categoria que e
      JTF_numerocam.setText("");
      JTF_infoPeli.setText("");
      JB_agregar.setEnabled(false);

      if(meterPeli.getEstado().equalsIgnoreCase("Estreno")==true)// si la peli es de estreno se encarece
        resTotal+=1800;
       else{
          resTotal+=1200;
       }//fin del else
```

Figura 3.32.Action Performed Alquilar Película

```
        pelis += meterPeli.getId()+"||";// dandole el id de la pelicula vamos añadiendo las pelis
        JTF_mont.setText(""+resTotal);// en el text field ponemos el importe total

      } //final del  boton agregar

  if(e.getSource()==JB_guardar){// la fuente del action event es guardar
      archivo();// llamamos al metodo archivo que nos metera los datos de la pelicula en el archivo
      /* despues de guardar en la base de datos(archivo) la película reseteamos todos los campos*/
      JTA_todasLasPelis.setText("");
      JTF_infoPeli.setText("");
      JTF_numerocam.setText("");
      peliculasJTA="";//el string vacio
      resTotal=0;
      pelis="";
      principal.setJdialog(false);// cogemos una instancia de la clase principal y llamamos al método setJdialog para que ya no este visible la pelic
  }//final del boton guardar

 if(e.getSource()==JB_cancelar){// la fuente del action event escancelar
     int seleccion = JOptionPane.showOptionDialog(JB_cancelar,"¿Quiere cancelar la seleccion de la película?","Cancelar",JOptionPane.YES_NO_CANCEL_OI
    if(seleccion==0){ // si cancelas reseteamos todos los campos.
        peliculasJTA="";
      JTA_todasLasPelis.setText("");
      JTF_numerocam.setText("");
      JTF_infoPeli.setText("");
      alquiler.setPeliculas("");
      principal.setJdialog(false);
      resTotal=0;
      pelis="";
    }//final del boton cancelar
 }
```

Figura 3.33.Action Performed Alquilar Película II

Dentro de este método, se tiene en cuenta la fecha de estreno de la película, para ello se ha importado la clase **GregorianCalendar**[3.3.3] mediante el cual, establecemos un sistema de fechas basado en el sistema actual gregoriano. Este sistema establece el número de días de cada mes del año incluyendo la excepcionalidad de los años bisiestos [3.3.4].

Como podemos apreciar en las figuras 3.32 y 3.33, dentro del método actionPerformed (Actionevent e) de la clase Seleccionpelicula.java, en función de la opción que el usuario quiera realizar en ese momento se tendrán diferentes casos:

- El primer caso se corresponde al pulsar sobre buscar película, en este caso lo único que hace la aplicación es comprobar si la película existe y si es así, se incluye la película en el campo de texto de abajo.

- El segundo caso se corresponde al botón agregar película, este caso sirve para incluir en el área de texto principal todas las películas y muestra el precio a pagar del total de ellas teniendo en cuenta que si la película es de estreno que tendríamos que pagar 1800 sino el precio se reduce a 1200.

- El tercer caso se corresponde al botón agregar. Este botón llama al método archivo (), el cual aparece en la Figura 3.34. Este método a su vez crea un archivo llamado seleccionPeliculas.txt que lo que incluirá será la cifra de coste del alquiler y los identificadores de cada película separados mediante el carácter ";". Por lo que ese alquiler quedará registrado en la base de datos de nuestra aplicación.

```
public void archivo(){

        try{
        File f0;
        f0 = new File("seleccionPeliculas.txt");
        FileOutputStream fos;
        fos=new FileOutputStream(f0);
        PrintStream prs;
        prs = new PrintStream(fos);



         prs.println(resTotal+";"+pelis);

    }//final del try
    catch(FileNotFoundException fnfe){
        JOptionPane.showMessageDialog(null,fnfe);
    }//final del catch


}
```

Figura 3.34.Archivo

- Por último, el último caso se corresponde con el botón cancelar, que nos lleva al panel anterior cancelándose el alquiler de la película.

3.3.3.2 Cierre Mensual de Alquileres

En la Figura 3.35 se muestra la ventana que aparece cuando el usuario dentro del menú movimientos selecciona el apartado de Cierre Mensual. El usuario en función del mes que seleccione se generará un informe con los datos correspondientes a ese mes.

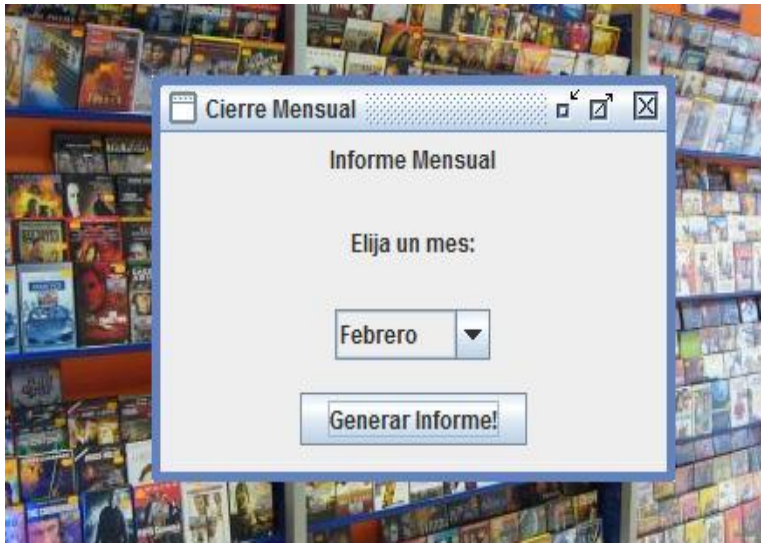


Figura 3.35.Cierre Mensual de Alquileres

Como se puede apreciar en la figura de arriba, la ventana es muy sencilla. Esta ventana consta de una lista desplegable (un Combobox), mediante la cual el usuario podrá elegir el mes sobre el que realizar el informe y un botón Generar Informe que generará el informe correspondiente. A continuación, una vez generado este informe, se aprecia cómo se detallan todos los movimientos realizados cada mes y el total de ingresos que se tienen para ese mes.

En la Figura 3.36 se muestra un ejemplo de un informe que la aplicación ha generado para el mes de Febrero. Como se puede comprobar, el total de movimientos efectuados en este mes son dos, cada uno de dichos movimientos se puede ver detalladamente el cliente que realizó el movimiento, el número de películas que se alquilaron, las fechas y el pago. También se aprecia el total de ingresos que tenemos para este mes.

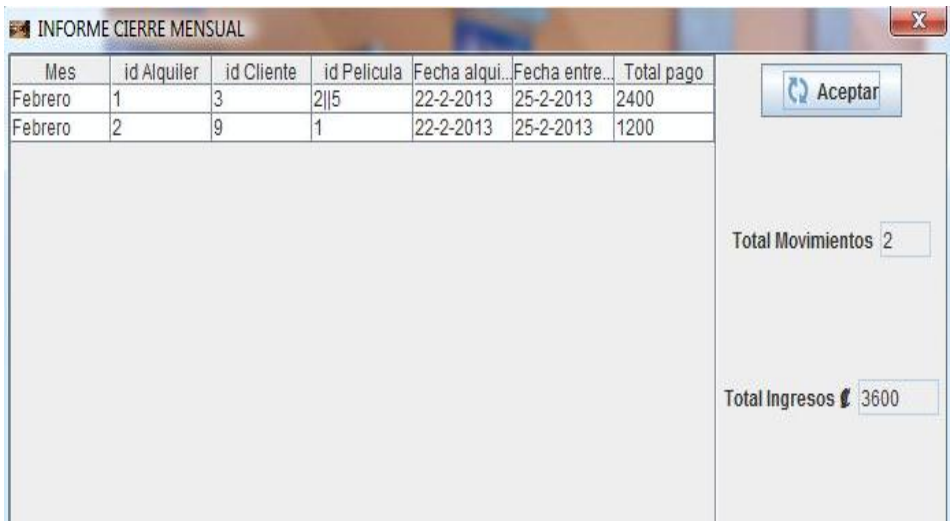


Figura 3.36.Informe Cierre Mensual

En la figura de arriba tenemos dos campos en la parte derecha: Total Movimientos y Total Ingresos. A continuación en la Figura 3.37 se muestra el código implementado de los métodos **escribeTotalResgistro ()** y **escribeTotalIngresos(),** que escriben en dos ficheros tanto el número total de movimientos como el total de los ingresos.

```
    public void escribeTotalRegistros(){
    try{

       File f4;
       f4= new File("temp_contadorRegistros.txt");
       FileOutputStream fos4;
       fos4 = new FileOutputStream(f4,true);
       PrintStream ps;
       ps = new PrintStream(fos4);
       String contadorDeRegistrosString;
       contadorDeRegistrosString = String.valueOf(contadorDeRegistros);
       ps.println(contadorDeRegistrosString);
    }//final del try
      catch(FileNotFoundException fnfe){
       JOptionPane.showMessageDialog(null, fnfe);
       }//final del catch

    }//final del metodo escribeTotalRegistros
public void escribeTotalIngresos(){

     try{
     File f5;
     f5=new File("temp_totalIngresos.txt");
     FileOutputStream fos5;
     fos5 = new FileOutputStream(f5);
     PrintStream ps5;
     ps5= new PrintStream(fos5);
     String totalIngresos;
     totalIngresos = String.valueOf(totalIngresosPorMes);
     ps5.println(totalIngresos);
     }//final del try
```

Figura 3.37.Métodos Informe Cierre Mensual

El método ***escribeTotalRegistros()*** muestra el número de movimientos que ha habido en ese mes, es decir, el número de alquileres independientemente de que en un alquiler el cliente haya podido alquilar una o varias películas, para eso se crea un archivo llamado temp_contadorRegistros.txt que va a contener un número, con el número de movimientos ocurridos en ese mes.

El método ***escribeTotalIngresos ()***, nos muestra la cantidad de ingresos generados por todos los alquileres en el mes en cuestión. Como en el método anterior, tenemos un archivo llamado temp_totalIngresos.txt en el que almacenamos la cantidad.

3.3.3.3 Recomendación de películas

En este apartado la aplicación recomienda una serie de películas en función de una serie de características del usuario. El usuario mediante un cuestionario manda a la aplicación una serie de características y mediante un algoritmo la aplicación le muestra al usuario distintas películas en función de sus gustos.

Esta parte de la aplicación está inspirada en la **televisión a la carta** o **Smart tv**. En base a lo que se ha explicado, nuestra aplicación tiene una función que permite al usuario en función de sus gustos personales tener la posibilidad de escoger una película a su gusto. En la Figura 3.38 se muestra el cuestionario que la aplicación ofrece al usuario para que en función de esos datos pueda generar un informe con las películas que recomiende al usuario.

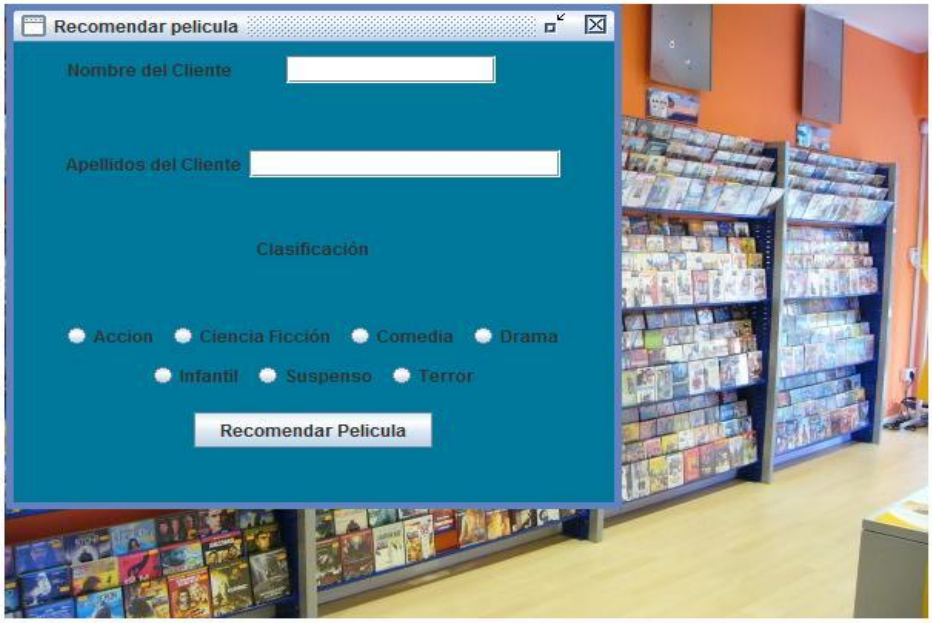


Figura 3.38.Recomendar Película

Dentro de la ventana que aparece en la figura de arriba el usuario introduce sus datos en JTextArea (áreas de texto) y en función de sus gustos personales elige el tipo de películas que prefiera mediante un conjunto de RadioButton.

A continuación el usuario pulsa en el botón recomendar película y aparece en la pantalla la recomendación que nos da la aplicación adecuada a sus gustos personales. En la Figura 3.39 se muestra la implementación de este apartado de Recomendar Película

```
try{

if(e.getSource()== JB_recomendarPelicula){//si la fuente del action event es insertar pelicula
    //obtenemos los datos de los campos de texto para que sean procesados..
      setNombre(JTF_nombreCliente.getText());
      setApellidos(JTF_apellidosCliente.getText());
      System.out.println("Hola: " + getNombre() + " " + getApellidos());
      if(getClasificacion().equals("accion")==true){
          LecturaXML lec = new LecturaXML();
          peliculasAccion = lec.devuelvePeliculasAccion();
      }
      else if(getClasificacion().equals("cienciaficcion")==true){
          LecturaXML lec = new LecturaXML();
          peliculasCienciaFiccion = lec.devuelvePeliculasCienciaFiccion();
      }
      else if(getClasificacion().equals("comedia")==true){
          LecturaXML lec = new LecturaXML();
          peliculasComedia = lec.devuelvePeliculasComedia();
      }
      else if(getClasificacion().equals("drama")==true){
          LecturaXML lec = new LecturaXML();
```

Figura 3.39.ImplementaciónRecomendar Película

Como se puede apreciar en el código implementado, en función del tipo de película que elija el usuario acorde a su clasificación se crea un objeto de la clase LecturaXML, que llamará a un método de esa clase que nos devolverá las películas que nos recomendara la aplicación en función de la categoría que hayamos elegido.

La clase LecturaXML recorrerá el archivo xml (películas.xml) y en función del género de la película que le guste más al cliente se le devolverá una recomendación de tres películas de ese género, las cuales el usuario podría alquilar o no posteriormente.

En la Figura 3.40 se muestra la estructura del archivo películas.xml. Como se puede apreciar este archivo está compuesto de una serie de etiquetas o nodos ordenados jerárquicamente que posibilitan que la clase LecturaXML los pueda recorrer y obtener uno de esos nodos o etiquetas.

```xml
<?xml version="1.0" encoding="UTF-8" ?>

<peliculas>
    <accion>
        <primera>Django</primera>
        <segunda>IronMan 3</segunda>
        <tercera>Gangster</tercera>
    </accion>
    <cienciaficcion>
        <primera>Avatar</primera>
        <segunda>Riddick</segunda>
        <tercera>Star Treck 2</tercera>
    </cienciaficcion>
    <comedia>
        <primera>Resacon en las Vegas</primera>
        <segunda>American Pie</segunda>
        <tercera>El Oso Ted</tercera>
    </comedia>
    <drama>
        <primera>Lincoln</primera>
        <segunda>Así Somos</segunda>
        <tercera>El Vuelo</tercera>
    </drama>
    <infantil>
        <primera>Buscando a Nemo</primera>
        <segunda>Monstruos University</segunda>
        <tercera>Los Pitufos 2</tercera>
    </infantil>
    <suspense>
        <primera>Destino Final</primera>
        <segunda>Asuntos Pendientes</segunda>
```

Figura 3.40.Archivo peliculas.xml

En la Figura 3.41 se puede apreciar el código que hemos usado para el método devuelvePeliculasDrama.

```java
public String[] devuelvePeliculasDrama(){


    try {
       DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
       DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
       Document doc = dBuilder.parse(new File("src/videoclub1/peliculas.xml"));
       doc.getDocumentElement().normalize();

       System.out.println("El elemento raiz es: " + doc.getDocumentElement().getNodeName());
       NodeList listaPeliculasDrama = doc.getElementsByTagName("drama");

       for (int i = 0; i < listaPeliculasDrama.getLength(); i ++) {

         Node drama = listaPeliculasDrama.item(i);

         if (drama.getNodeType() == Node.ELEMENT_NODE) {

         Element elemento = (Element) drama;

         peliculasDrama = new String[3];
         peliculasDrama[0] = getTagValue("primera", elemento);
         peliculasDrama[1] = getTagValue("segunda", elemento);
         peliculasDrama[2] = getTagValue("tercera", elemento);
```

Figura 3.41.Implementación devuelve Película

En la figura de arriba tenemos el ejemplo en el caso que los gustos personales del usuario sean las películas de drama.

Como se ha citado anteriormente, se crea un objeto de la clase LecturaXML que llamará al método devuelvePeliculasDrama() que nos devolverá las películas recomendadas por la aplicación basada en los gustos del usuario, en este caso nos devuelve las de tipo drama.

Este método recorre el xml citado anteriormente "películas.xml" de manera que cuando alcance el nodo o etiqueta con el valor drama, almacena en un array los tres elementos que completan su estructura de nodos. Cada etiqueta o nodo de cada género cinematográfico contiene tres etiquetas llamadas primera, segunda y tercera que son las tres opciones de ese género cinematográfico que se ofrecen de recomendación al usuario.

3.3.3.4 Ver Cartelera

En este último apartado del menú movimientos, la aplicación tiene una opción llamada VerCartelera mediante la cual al pinchar en este submenú se puede acceder a una página web donde se puede ver la cartelera actual que hay en el cine. En la Figura 3.42 se muestra la implementación de la clase VerCartelera.

```
package videoclub1;

import javax.swing.JInternalFrame;
import javax.swing.JOptionPane;

/**
 *
 * @author efren
 */
public class VerCartelera extends  JInternalFrame{



    public VerCartelera(){


        String direccion = "http://www.ecartelera.com";
            try
             {
                Runtime.getRuntime().exec("\"C:\\Program Files\\Internet Explorer\\IEXPLORE.EXE\" " + direccion);
             }
            catch(Exception err)
             {
                JOptionPane.showMessageDialog(null,"Error: "+err);
             }

    }
  }
```

Figura 3.42.VerCartelera

Para este apartado como se puede apreciar en la figura de arriba se utiliza la clase Runtime [3.3.5] que se utiliza para ejecutar la URL que se le pase por parámetro.

3.3.4 Listados

En este apartado se va a detallar el menú listados, el cual consta de varios submenús cada uno de los cuales con una función distinta. El menú listado es el último de los cuatro menús de la aplicación. A continuación vamos a explicar cada uno de ellos.

- **Listado de todas las películas**: este apartado muestra una tabla en la que aparece la lista de todas las películas de nuestra aplicación, se incluye el identificador de película, el nombre, la clasificación, el reparto, el año de estreno y el estado.

- **Listado de todos los clientes**: este otro apartado muestra una tabla que contiene los clientes de la base de datos de la aplicación, ordenados según su identificador, su nombre, su dirección, su e-mail, su teléfono y su DNI o pasaporte.

- **Listado de las películas por cliente**: en este apartado se puede apreciar un panel dividido en dos partes, en la parte izquierda el usuario puede seleccionar el identificador del cliente para mostrar en la parte derecha de la tabla el numero de alquiler que ha realizado y las alquiladas por dicho usuario.

- **Día del mes con el alquiler más alto y más bajo**: este último apartado del menú listados se compone de una pequeña ventana con un panel, el cual seleccionando un mes, indica el día de dicho mes en el que se realizo el alquiler más alto hasta la fecha y su correspondiente cantidad que corresponderá a los ingresos del día, así como el día en el que se realizo el alquiler más bajo hasta la fecha y su cuantía económica.

3.3.4.1 Listado de todas las películas

En la Figura 3.43 se muestra un listado de todas las películas que están en la base de datos de la aplicación de este Proyecto Fin de Carrera.

Figura 3.43.Listado de todas las peliculas

Como se puede apreciar, el usuario al pulsar sobre el submenú listado de todas las películas se encuentra una pantalla tal que así, en la que se puede ver todas las películas que hay divididas en función de su ID, que será su identificador de película. Ese ID como se ha citado anteriormente es el que se le muestra a la aplicación a la hora de realizar el alquiler.

También se encuentran las películas divididas por su nombre, su clasificación (drama, drama histórico, aventura, suspense, acción, etc.), según su reparto, año de estreno y si actualmente es una película de estreno o no.

3.3.4.2 Listado de todos los clientes

En la Figura 3.44 se muestra un listado de todos los clientes que están en la base de datos de la aplicación de este Proyecto Fin de Carrera.

Figura 3.44.Listado de todos los clientes

Como se puede observar en la imagen anterior, cuando el usuario selecciona el submenú listado de todos los clientes, la aplicación abre una ventana en la que se encuentra insertada una tabla con la lista de clientes de la aplicación divididos por su ID además de, su nombre, dirección, email, teléfono, DNI o pasaporte.

En la Figura 3.45 se puede apreciar el código implementado para el constructor de la clase Listado Clientes.

```
public ListadoClientes(){

    super("Lista de los Clientes", true, true , true , true);
    setVisible(true);
    setLocation(420, 220);
    setSize(575,300);
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);

    String encabezados[]= {"ID", "Nombre", " Direccion", "e-mail", "Telefono", "DNI o pasaporte"};


    String listadoDeClientes[][] = getDatosArchivos();
    JT_tablaDeClientes = new JTable();
    JScrollPane JSP_barraDeslizante= new JScrollPane(JT_tablaDeClientes);

    DTM_modeloDeTabla = new DefaultTableModel(listadoDeClientes, encabezados);//pasamos los argumentos a
    //el array y el encabezado de la tabla.

    JT_tablaDeClientes.setModel(DTM_modeloDeTabla);
    JT_tablaDeClientes.getTableHeader().setResizingAllowed(false);///no se puede cambiar el tamaño a la
    JT_tablaDeClientes.getTableHeader().setReorderingAllowed(false);///no se puede cambiar la posicion d
    JT_tablaDeClientes.setEnabled(true);
    this.add(JSP_barraDeslizante);


}/* fin del constructor*/
```

Figura 3.45.Implementación Listado Clientes

Como se puede apreciar en la figura de arriba, en este trozo de código se va a llamar a un método llamado getDatosArchivos() que se explica a continuación, en el que básicamente lo que se obtiene son cuantos clientes hay en la base de datos de la aplicación y los datos de cada cliente. Este código es muy similar al listado de todas las películas, la forma de obtener los datos para ingresarlos en la tabla es la misma.

En la Figura 3.46 se muestra la implementación del método getDatosArchivos(). En este método lo que se obtiene es el número de clientes y los datos de cada uno de ellos. Con el método getTodosLosClientes() se obtienen el número de clientes de los que consta nuestra aplicación y para cada uno de ellos se irán obteniendo diferentes datos como el identificador, el nombre, los apellidos, la dirección, el teléfono, el email y el DNI o pasaporte.

```
public String [][] getDatosArchivos(){

    insertCli= new insertarClientes(null, null, null, null, null, null);
    String datosClientes[][] = new String[insertCli.cantidadClientes()][6];//creamos el string con el nu
    //cantidadClientes() de la clase insertarClientes a traves de la instancia de la clase insertCli.

    int contador=0;

    int  ID=0, NOMBRE=1, TELEFONO=2, DIRECCION=3, EMAIL=4, DNIPASP=5;

    insertarClientes arregloClientes[] = insertCli.getTodosLosClientes();//llamamos al metodo getTodosLo
    for(int i=0; i<arregloClientes.length;i++){
    insertCli = arregloClientes[i];
    datosClientes[contador][ID]= insertCli.getId();
    datosClientes[contador][NOMBRE]=  insertCli.getNombre();
    datosClientes[contador][TELEFONO]= ""+ insertCli.getTelefono();
    datosClientes[contador][DIRECCION]= ""+ insertCli.getDireccion();
    datosClientes[contador][EMAIL]= ""+ insertCli.getEmail();
    datosClientes[contador][DNIPASP]= ""+ insertCli.getDniPasp();
    contador++;
    }

    return datosClientes;
}//final del metodo getDatosArchivos
```

Figura 3.46.getDatosArchivos

3.3.4.3 Listado de las películas por cliente

En la Figura 3.47 se muestra el listado de películas por cliente.

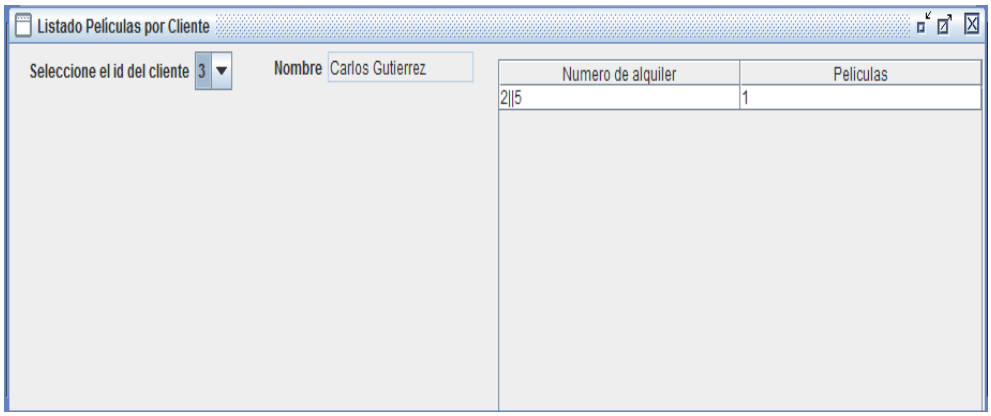


Figura 3.47.Listado Películas por Cliente

Una vez el usuario selecciona el submenú listado de películas por cliente, se mostrará una ventana dividida en dos partes. En la parte izquierda de la ventana se puede apreciar un comboBox mediante el cual el usuario puede elegir el identificador de cliente, que posteriormente se asociará con su nombre y su apellido.

En la parte derecha de la ventana se tiene una tabla con dos columnas: número de alquiler y películas. Para cada cliente, esta tabla se irá rellenando con el número de alquileres que haya hecho este cliente en cuestión y el número de películas que ha alquilado.

El desarrollo de este apartado es un poco complejo. La tabla de la derecha se completa gracias al rellenado de una matriz que será un array de filas y columnas. En las Figuras 3.48 y 3.49 se puede apreciar el código implementado para el constructor de la clase Listado PeliculasClientes.

```
public listadoPeliculasCliente(){

    super("Listado Películas por Cliente", true, true , true , true);
    setVisible(true);
    setLocation(100, 10);
    setSize(920,270);//tamaño de la ventana
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);

    //creamos un string de tamaño la cuenta de clientes
    clientesID = new String[CuentaClientes()];

    llenadoDelVector();

    //instanciamos los paneles
    JP_uno = new JPanel();
    JP_uno.setLayout(new GridLayout(1,1));
    JP_dos = new JPanel();
    JP_comboBox = new JPanel();
    JP_datosCliente = new JPanel();
    JP_tabla = new JPanel();

    //pegamos los subpaneles en los paneles

    JP_uno.add(JP_comboBox);
    JP_uno.add(JP_datosCliente);
    JP_dos.add(JP_tabla);
```

Figura 3.48.Implementación Películas por Cliente

```
JCB_Clientes = new JComboBox(clientesID);
JP_comboBox.add(new JLabel("Seleccione el id del cliente"));
JP_comboBox.add(JCB_Clientes);//el comboBox tendra de tamaño el numero de id de todos los clientes
JCB_Clientes.addActionListener(this);

//panel con el nombre del cliente

JP_datosCliente.add(new JLabel("Nombre"));
JTF_nombres= new JTextField(12);
JP_datosCliente.add(JTF_nombres);
JTF_nombres.setEditable(false);



//construimos el panel principal

JP_principal = new JPanel();
JP_principal.setLayout(new GridLayout(1,2));
JP_principal.add(JP_uno);
JP_principal.add(JP_dos);

JT_tabla = new JTable();
JScrollPane JSP_barra=new JScrollPane(JT_tabla);


 String matriz[][]=new String[1][1];
 DTM_tabla = new DefaultTableModel(matriz,encabezados);
 JT_tabla.setModel(DTM_tabla);
 JT_tabla.getTableHeader().setResizingAllowed(true);//no se puede cambiar el tamaño a la tabla
 JT_tabla.getTableHeader().setReorderingAllowed(false);//no se puede cambiar la posicion de las filas en la tabla
 JT_tabla.setEnabled(false);
```

Figura 3.49.Implementación Películas por Cliente 2

Como se puede apreciar en las figuras de arriba, se crean dos paneles. El panel de la izquierda constara de un comboBox donde el usuario selecciona el identificador de cliente y una etiqueta donde se escribe el nombre y apellidos del cliente. El panel de la derecha constara de una tabla donde se insertan los datos correspondientes al número de alquiler y la película alquilada de cada cliente. Estos dos paneles son añadidos al panel principal.

3.3.4.4 Día del alquiler más alto y más bajo

En la Figura 3.50 se muestra la ventana que aparece cuando el usuario selecciona el submenú día con el alquiler Alto-Bajo.

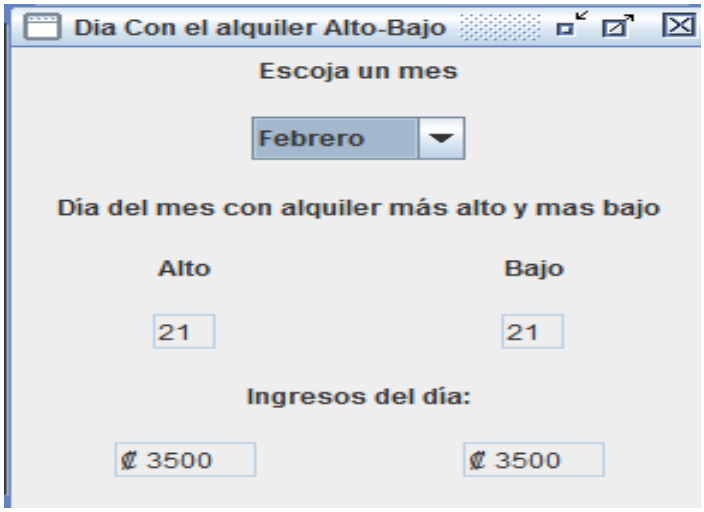


Figura 3.50.Día con el alquiler Alto-Bajo

Como se puede apreciar en la figura de arriba, al seleccionar esta opción aparece una ventana en la que para cada mes del año tendremos un balance de dicho mes. En dicho balance aparece el día del mes con el alquiler más alto, el día del mes con el alquiler más bajo y los ingresos de esos días.

CAPÍTULO 4. Evaluación de la aplicación

En este capítulo se describe la evaluación preliminar que se ha llevado a cabo de la aplicación del videoclub virtual desarrollado para el Proyecto Final de Carrera. Para ello, se ha desarrollado un cuestionario que recoge la valoración subjetiva de los usuarios que han utilizado la aplicación. Estos resultados parten sobre la base de un número no muy grande de personas pero suficiente como para poder darnos cuenta de los aspectos a mejorar de nuestra aplicación y hacer una adecuada y completa evaluación de la aplicación. Los resultados obtenidos se muestran a lo largo de este capítulo en forma de gráficas.

4.1Metodología de la evaluación

La evaluación y análisis de los datos sobre esta aplicación es una parte fundamental del trabajo para mejorar las prestaciones en versiones sucesivas de la aplicación.

Esta aplicación se ha llevado a cabo a través de valoraciones de calidad. Para ello, se ha realizado un cuestionario que recoge la opinión subjetiva y el grado de satisfacción de los usuarios, obteniendo así una evaluación cualitativa de la percepción del sistema por parte de los usuarios.

Con esta técnica de análisis se pretende además obtener sin un consumo excesivo de recursos, información básica acerca de la opinión de los potenciales usuarios de la aplicación. En total, se ha requerido la colaboración de 20 usuarios para cubrir todas las posibles alternativas de usuarios de la aplicación que puedan existir.

Los aspectos que se han querido analizar son: el grado en el cual el usuario valora como de manejable es la aplicación en sí, la velocidad percibida de la de interacción, la presencia de errores, la utilidad de la aplicación y el nivel de satisfacción con el sistema global. Además, información adicional de los usuarios sobre su grado de conocimiento acerca de nuevas tecnologías sirven para hacerse una idea del perfil de estos usuarios.

El cuestionario elaborado para este fin consta de 5 preguntas (Figura 4.1). Cada pregunta tiene 4 posibles respuestas (Muy buena, buena, regular y mala) y solo se puede elegir una. Para proceder a rellenar este cuestionario, cada usuario ha hecho uso de la aplicación por un tiempo de 25 minutos, tiempo más que suficiente para probarla con total libertad para descubrir las distintas funcionalidades que ofrece la aplicación

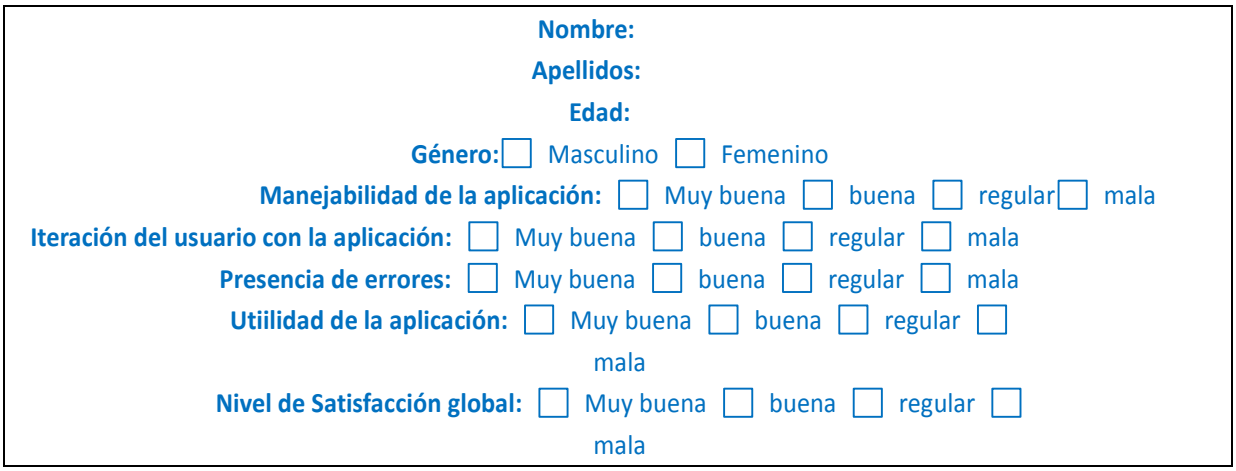


Figura 4.1.Cuestionario evaluación

A partir de estas respuestas que dé el usuario evaluando los cinco aspectos claves de la aplicación, se han obtenido las estadísticas para el análisis de los resultados que vamos a explicar a continuación mediante unas gráficas.

4.2 Resultados de la evaluación

Como se ha comentado anteriormente, el test de evaluación de la aplicación ha sido realizado a 20 usuarios a los cuales se les ha permitido navegar por la aplicación para que ellos mismos comprueben las funcionalidades de la aplicación.
A continuación en las sucesivas figuras se presenta el resultado de la evaluación de la aplicación en función de ciertas características.

Además para cada uno de los aspectos sobre los que se pregunta al usuario se incluye una tabla con el valor medio, el valor máximo, la mediana, el valor mínimo y la varianza. Para llevar a cabo estos cálculos se puntúa de 1 a 4 cada cuestión, siendo 1 en el caso de que la respuesta sea "Mala", siendo 2 en el caso de "Regular", siendo 3 en el caso de "Buena" y siendo 4 si la

respuesta es “Muy Buena”.

Pregunta 1. Puntúe en una escala del 1 al 4 como de manejable le parece la aplicación

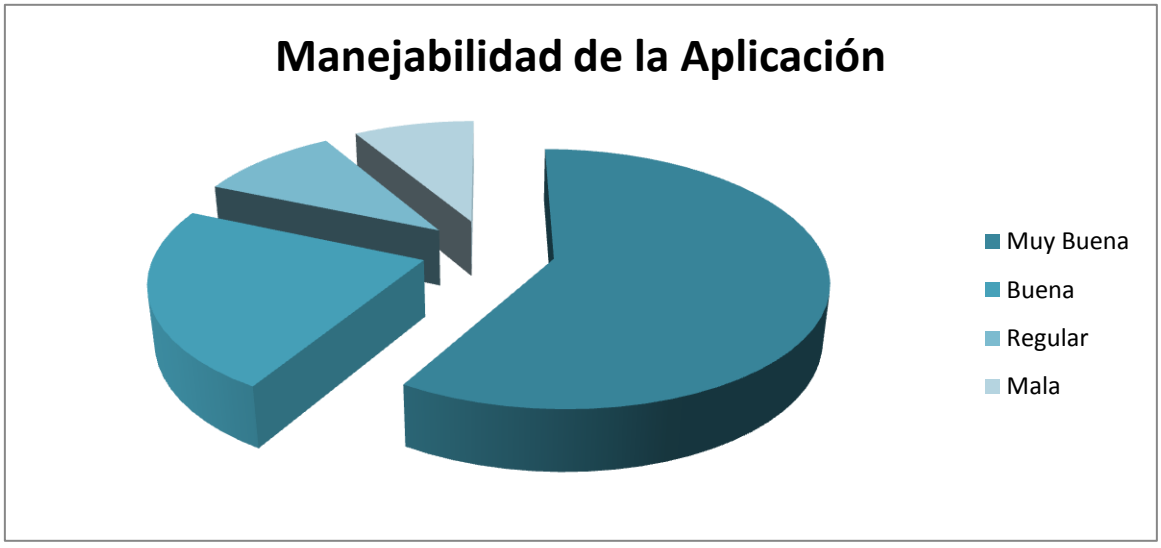


Figura 4.2. Manejabilidad de la aplicación

La Tabla 4.1 muestra el valor medio, valor máximo, valor mínimo, mediana y varianza para la pregunta sobre la manejabilidad de la aplicación.

Valor Medio	Mediana	Valor Maximo	Valor Mínimo	Varianza
3,25	4	4	1	1,039

Tabla 4.1: valores calculados para la manejabilidad de la aplicación.

Como vemos, la mayor parte de los encuestados creen que la aplicación es muy manejable y tan solo una parte considera que la manejabilidad de la aplicación es regular o mala, por lo que en este aspecto el resultado es bastante positivo.

Pregunta 2. Puntúe en una escala del 1 al 4 la velocidad de la aplicación

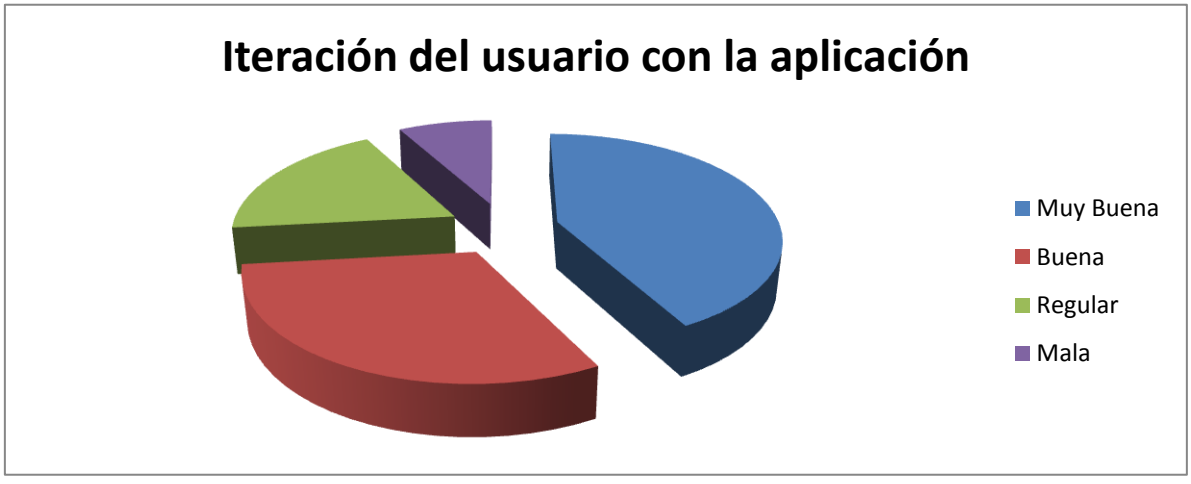


Figura 4.3. Iteración del usuario con la aplicación

La Tabla 4.2 muestra el valor medio, valor máximo, valor mínimo, mediana y varianza para la pregunta sobre la iteración del usuario con la aplicación.

Valor Medio	Mediana	Valor Maximo	Valor Mínimo	Varianza
3	3	4	1	0,947

Tabla 4.2: valores calculados para la velocidad de la aplicación.

Como se puede apreciar, respecto a la iteración del usuario con la aplicación baja el número de encuestados que la definen como muy buena, aun así, tres cuartas partes de los encuestados considera que es buena o muy buena, de manera que el resultado sigue siendo positivo en este aspecto aunque habrá que tenerlo en cuenta para futuras versiones.

Pregunta 3. Puntúe en una escala del 1 al 4 la presencia de errores en la aplicación

Figura 4.4. Presencia de errores

La Tabla 4.3 muestra el valor medio, valor máximo, valor mínimo, mediana y varianza para la pregunta sobre la presencia de errores en aplicación.

Valor Medio	Mediana	Valor Maximo	Valor Mínimo	Varianza
2,75	3	4	1	1,1447

Tabla 4.2.3: presencia de errores en la aplicación.

Tal y como se refleja en la figura de arriba un número importante de usuarios considera que la aplicación no está totalmente pulida y consta con la presencia de algunos errores, cosa normal

teniendo en cuenta que es una primera versión y estos errores irán desapareciendo en versiones futuras aun así más de la mitad de los encuestados cree que la presencia de errores es buena o muy buena, por lo que se tendrá en cuenta para versiones posteriores tal y como expondremos posteriormente en el apartado de conclusiones y trabajo futuro.

Pregunta 4. Puntúe en una escala del 1 al 4 la utilidad de la aplicación

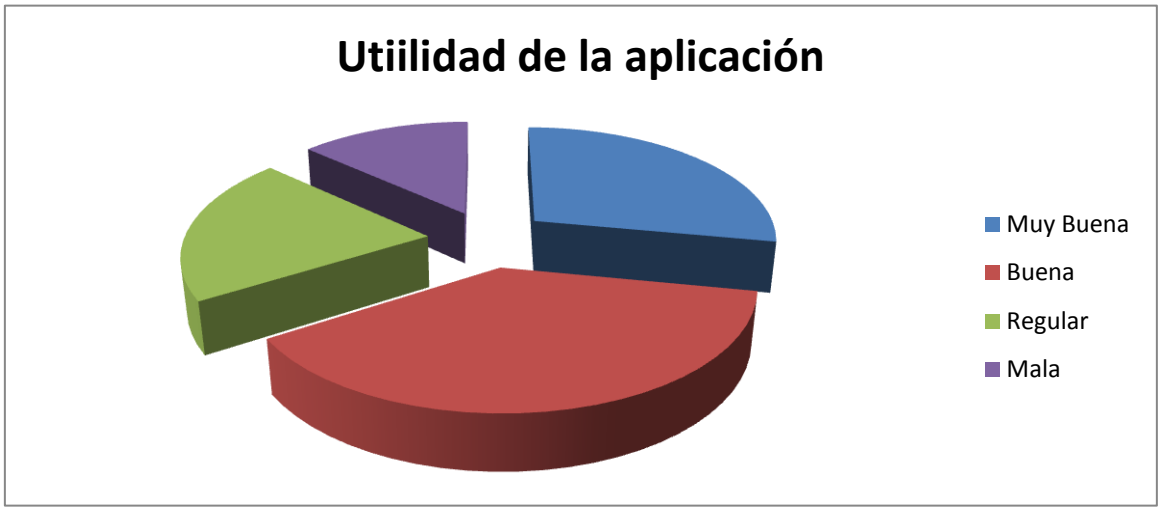


Figura 4.5. Utilidad de la aplicación

La Tabla 4.2.4 muestra el valor medio, valor máximo, valor mínimo, mediana y varianza para la pregunta sobre la utilidad de la aplicación.

Valor Medio	Mediana	Valor Maximo	Valor Mínimo	Varianza
2,85	3	4	1	0,871

Tabla 4.4: utilidad de la aplicación.

Como se aprecia en la figura anterior, en cuanto a la utilidad de la aplicación, la mayoría de los encuestados apuestan por la opción buena o muy buena, aunque sorprende ver como el número de personas que se deciden por la opción buena es superior al número de personas que se deciden por la opción muy buena, esta opción está casi empatada con la del número de personas que deciden la opción regular, lo que nos muestra que el usuario todavía no ve al 100% si puede ser del todo útil esta aplicación, algo que comentaremos mas tarde en el trabajo futuro y las conclusiones.

Pregunta 5. Puntue en una escala del 1 al 4 el nivel de satisfaccion global de la aplicación

Figura 4.6. Nivel de Satisfacción global

La Tabla 4.2.5 muestra el valor medio, valor máximo, valor mínimo, mediana y varianza para la pregunta sobre el nivel de satisfacción global de la aplicación.

Valor Medio	Mediana	Valor Maximo	Valor Mínimo	Varianza
2,85	3	4	1	0,871

Tabla 4.5: nivel de satisfacción global de la aplicación.

El nivel de satisfacción global es muy bueno o bueno para la gran mayoría de los usuarios, por lo que en el termino general podemos decir que el trabajo ha sido exitoso, aun así en futuras mejoras de la aplicación insistiremos en mejorar algunos aspectos en los cuales los usuarios han incidido especialmente no valorándolos todo lo bien que pensábamos.

CAPÍTULO 5. Conclusiones y trabajo futuro

En este capítulo se hace un balance del trabajo realizado en el presente Proyecto Final de Carrera. Partiendo de la revisión de los resultados logrados, se comprueba que los objetivos fijados inicialmente han sido cumplidos y se exponen las conclusiones obtenidas. Para finalizar, se presentan los posibles trabajos futuros que se podrían desarrollar sobre la aplicación, de forma que aumentará su eficiencia y el número de funcionalidades.

Sobre la utilidad de la aplicación y la presencia de errores como hemos comentado en el apartado anterior trataremos de mejorar esas dos características de la aplicación, de manera que en versiones futuras corrijamos estos aspectos y le demos al usuario mayor satisfacción global.

5.1Conclusiones

En este Proyecto Fin de carrera se ha desarrollado una aplicación basada en un videoclub virtual en el que el usuario pueda alquilar una película desde un ordenador o un dispositivo móvil. Se trata de una aplicación que permite a los usuarios registrarse como usuario en la aplicación, alquilar una película, poder rellenar un cuestionario sobre sus gustos de manera que la aplicación le recomiende películas, poder ver un listado de todos los clientes, películas o de los movimientos acontecidos en un mes.

Los servicios que ofrece la aplicación se dividen en menús, en función de lo que quiera hacer el usuario podrá acceder a uno u otro. Desde el menú **Sistema** el usuario podrá acceder a una serie de datos genéricos sobre el usuario, el equipo, el sistema operativo, el procesador y la versión desde donde lanzamos la aplicación.

Desde el menú **Actualiza** el usuario accede a numerosas funcionalidades como ingresar un cliente nuevo en la base de datos del videoclub, ingresar una película nueva en la base de datos, borrar una película o un cliente de la base de datos, actualizar una película o un cliente de

manera que si anteriormente se metieron mal los datos o el cliente registra un cambio de domicilio, se pueda reflejar en la base de datos.

El menú **Actualiza** se es imprescindible para la aplicación, ya que para realizar un alquiler o consultar los movimientos acontecidos en el último mes, el usuario deberá crear una base de datos con los clientes y películas que conste el videoclub.

Otro menú que aparece en la aplicación es el menú **Movimientos**, este menú es el núcleo central sobre el que gira la aplicación debido a que la parte más importante, en este caso, el alquiler de la película por parte del usuario, se realiza desde este menú. Además de realizar el alquiler, desde el menú **Movimientos** generaremos los informes de cada mes, podremos acceder a la cartelera y disfrutaremos de una serie de recomendaciones que nos ofrecerá la aplicación en base a nuestros gustos personales.

Por último, nos encontramos con el menú **Listados**, a través de este menú el usuario dispondrá de una lista con todos los clientes de la aplicación, otra lista con todas las películas, podrá conocer las películas alquiladas por un cliente u otro, el día del mes que se realizo el alquiler más alto o más bajo. Este menú nos ofrece una serie de estadísticas muy útiles para conocer como se está usando la aplicación.

En el proceso de desarrollo de este proyecto he disfrutado con todos y cada uno de los retos que se me han ido presentando. Todo ello gracias al afán de superación y a algunas palabras de aliento, las cuales han logrado que las soluciones solventasen cada uno de los retos que tenía por delante, además de perseverar en cada uno de los retos, he podido aumentar mis conocimientos sobre los lenguajes de programación, los dispositivos móviles, las nuevas tecnologías y como se presenta el mercado actual ante la necesidad de unos usuarios cada vez con más conocimientos y mayor necesidad de recibir una atención personalizada.

Todo ello hace que este Proyecto Fin de Carrera no me haya servido para cumplir expediente sino que he podido comprobar hacia donde tenderemos en el futuro en el mundo de las nuevas tecnologías y las aplicaciones informáticas. También he aprendido como cada vez, tendemos a el uso del móvil para llevar a cabo funciones que antes solo podíamos hacer con una computadora

La creación de la aplicación en Java, me ha enseñado a manejar mejor bases de datos, aspectos gráficos y funcionales, aportándome nociones imprescindibles para el futuro laboral de un buen desarrollador.

5.2 Trabajo futuro

Tras la finalización de este proyecto, se detallan una serie de puntos adicionales sobre los que se podría trabajar en un futuro con el fin de aumentar las prestaciones de la aplicación.

Por un lado se incluirían ampliaciones de funcionalidades ya existentes con el fin de mejorar el rendimiento de la aplicación, las cuales se podrían incorporar y por otro se añadirían funcionalidades globales del sistema, con las que se quieren cubrir aspectos que con la versión actual no se realizan.

Se detallan a continuación los puntos adicionales sobre los que se podría trabajar para obtener mayores prestaciones en la aplicación desarrollada

Ampliaciones de funcionalidades ya existentes:

- Ampliar la base de datos para las recomendaciones de las peliculas, de manera que el programa pueda contar con mucha mas información acerca del usuario y de esta manera seleccionar una o varias peliculas en funcion a muchos mas parametros.

- Adaptar la aplicación para que pueda ser usada en un dispositivo móvil y no se limite solamente a una aplicación de escritorio.

- Ampliar el menu Listados de manera que no solo se puedan generar informes con todos los movimientos en nuestro videoclub por mes sino tambien por trimestre o año.

Adición de funcionalidades globales al sistema:

- Crear una señal de aviso de manera que cuando un cliente no ha devuelto a tiempo la película, se incremente el precio de manera directamente proporcional al tiempo que tarde en devolverla.

- Tener la opción de comprar una película y no solo se alquilarla

- Crear la opción de suscripción mensual basada en aplicaciones existentes en el mercado, de manera que se puedan acceder a todos los contenidos.

Glosario

- **Android:** es un sistema operativo basado en Linux, diseñado principalmente para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tablets.

- **API(Application Programming Interface):** es un conjunto de funciones y procedimientos que provee un sistema operativo, una aplicación o una biblioteca y que permiten leer o añadir contenido de un software en sus sitios web.

- **BBDD(Data Base):** una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

- **BSD(Berkeley Software Distribution):** es un sistema operativo derivado del sistema Unix nacido a partir de los aportes realizados a ese sistema por la universidad de California en Berkeley.

- **HTML5:** es la quinta revisión del lenguaje de marcado HTML. Esta revisión especifica dos variantes de sintaxis para HTML: la clásica, conocida como HTML5 y otra variante, XHTML, conocida con la sintaxisXHTML5.

- **iOS:** es un sistema operativo móvil de la multinacional Apple Inc. Fue originariamente desarrollado para iPhone y posteriormente ha sido usado en dispositivos como el iPod Touch y el iPad.

- **Java:** es un lenguaje de programación desarrollado en 1995 por James Gosling de Sun Mycrosystems específicamente para entornos de red como Internet. Se trata de un lenguaje orientado a objetos, lo que significa que sus programas se construyen con modulos de código. Java a diferencia de otros lenguajes es interpretado, por lo que, tiene más tiempo de ejecución.

- **JDBC (Java Database Connectivity):** es una api que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

- **JDK (Java Development Kit):** es un software que provee herramientas de desarrollo para la creación de programas en Java.

- **JRE (Java Runtime Environment):** es un conjunto de utilidades que permiten la ejecución de programas java.

- **JVM (Java Virtual Machine):**es una maquina virtual de proceso nativo, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (bytecode) el cual es generado por el compilador del lenguaje Java.

- **MMU:** es un dispositivo de Hardware formado por un grupo de circuitos integrados, responsable del manejo de los accesos a la memoria por parte de la unidad de procesamiento central(CPU)

- **MySQL (My Structured Query Language):** es, multihilo y multiusuario con más de seis millones de instalaciones.

- **ODBC(Open Database Connectivity):** es un estándar de acceso a las bases de datos desarrollado por SQL Access Group en 1992, cuyo objetivo es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar que sistema de gestión de bases de datos (DBMS) almacene los datos.

- **OpenGL(Open Graphics Library):** es una aplicación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzca gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas tales como, puntos, líneas y triángulos.

- **PDA (Personal Digital Assistant):** es una computadora de mano diseñada como agenda personal electrónica de bolsillo.

- **RAM (Random Access Memory):** se utiliza como memoria de trabajo de computadoras para el sistema operativo, los programas y la mayor parte del software.

- **SDK (Software Development Kit):** es un kit de desarrollo de software. En el caso del SDK de Android, es el proceso por el cual nuevas aplicaciones se crean para este sistema operativo móvil. Incluye un conjunto de herramientas de desarrollo como bibliotecas, un depurador, un terminal emulador basado en QEMU, código de ejemplo, documentación.

- **SGBD:** es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos.

- **SMART TV:** la televisión inteligente describe la integración de internet y de las características Web 2.0 a la televisión digital así como la convergencia tecnológica entre los ordenadores y estos televisores y el STB.

- **URL(Uniform Resource Locator):** es un identificador de recursos uniforme cuyos recursos referidos pueden cambiar, la dirección puede apuntar a recursos variables con el tiempo. Están formados por una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que designa recursos en una red, como Internet.

- **USB(Universal Serial Bus):** es un estándar industrial desarrollado en 1990, que define los cables, conectores y protocolos usados en el bus para conectar, comunicar y proveer de alimentacion eléctrica entre ordenadores y periféricos y dispositivos electrónicos.

- **WWW(World Wide Web):** es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles via Internet.

- **XHTML(eXtensible Hypertext Markup Language):** es básicamente html expresado como XML válido. Es mas estricto a nivel técnico, pero esto permite que posteriormente sea mas fácil a hacer cambios o buscar errores entre otros.

- **XML(eXtensible Markup Language):** es un lenguaje de marcas desarrollado por W3C, utilizado para almacenar datos en forma legible.Se trata de una adaptación y simplificación del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar grandes documentos.

Bibliografía

[1.1.1] Programar en Java

H.M. Deitel. Cómo programar en Java. 1ª edición. México: Prentice Hall, 1998.

[1.2.1] Auge en el uso de aplicaciones móviles

Disponible:
http://www.einnova.com/notas/diseno-web/diseno-aplicacion-movil-comercio-electronico
[14 de enero de 2015]

[1.2.2] Uso de aplicaciones móviles

Disponible:
https://www.yeeply.com/blog/informe-sobre-el-uso-de-apps-en-espana-201/
[30 de septiembre de 2014]

[2.2.1.1] App Store

Disponible:
http://store.apple.com/es/

[2.2.1.2] Programar en android

Burnette, Ed. Programación Android. 2ª edición. Madrid: Anaya, 2012. ISBN: 978-84-415-2876-5

[2.2.1.3] Windows Market Place

Disponible:
http://www.windowsmarketplace.com

[2.2.1.4] Ovi Store

Disponible:
http://store.ovi.com/

[2.2.2.1] Mercado de Smartphones por SO

Disponible:
http://www.norbertogallego.com/mercado-de-smarphones-por-sistema-operativo/2010/10/06/

[2.2.3.1] Televisión a la carta

Disponible:
http://news.libertagia.com/technology/la-television-conectada-gana-terreno-en-espana-4305?lang=es
http://www.elmundo.es/television/2014/02/18/5303931422601d15258b457c.html

[2.2.3.2] Youzee

Disponible:
http://www.xataka.com/hogar-digital/youzee-diez-respuestas-sobre-su-funcionamiento/
[8 de febrero de 2012]

[2.2.3.3] Wuaki.tv

Disponible:
https://es.wuaki.tv/

[2.2.3.4] Cineclick

Disponible:
https://cineclick.com/peliculas

[2.2.3.5] Filmin

Disponible:
https://www.filmin.es/

[3.3.1] Clase String Tokenizer

Disponible:
http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/colecciones/stringtokenizer.htm

[3.3.2] Clase Mask Formatter

Disponible:
http://qmarqeva.wordpress.com/2009/06/15/jformattedtextfield-y-maskformatter/
[16 de junio de 2009]

[3.3.3] Clase Gregorian Calendar

Disponible:
http://www.java2s.com/Code/JavaAPI/java.util/GregorianCalendarisLeapYear.htm

[3.3.4] Método isLeapYear

Disponible:
http://www.dreamincode.net/forums/topic/19032-isleapyear/

[3.3.5] Método getRuntime()

Disponible:
http://www.tutorialspoint.com/java/lang/runtime_getruntime.htm
http://docs.oracle.com/javase/7/docs/api/java/lang/Runtime.html